REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE 20 October 1995	3. REPORT TYPE AND DATES COVERED Final; 1 June 1992 to 30 Sept. 1995

4. TITLE AND SUBTITLE

Protocol Engineering for Multimedia Communications

5. FUNDING NUMBERS

DAAL03-92-G-0184

6. AUTHOR(S)

Ming T. Liu

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Dept. of Computer and Information Science
The Ohio State University
2015 Neil Avenue
Columbus, OH 43210-1277

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

U.S. Army Research Office
P.O. Box 12211
Research Triangle Park, NC 27709-2211

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

The views, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other documentation.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The final summarizes the research results obtained, covering five areas in protocol engineering for multimedia communications. The five area are 1) protocol conversion, 2) conformance testing, 3) multimedia protocol design, 4) LANs extension, and 5) mobile communication. For each area of research, the problems under investigation are first described, and then the major results obtained are summarized. A total of 31 publications, including 5 Ph.D. dissertations, 7 journal publications and 19 publications in the proceedings of international conferences, have been credited to contract. In the appedix, 5 reprints that were not included in the previous technical reports, are included in the final report.

19960311 074

14. SUBJECT TERMS

Protocol enginnering, multimedia communications, protocol conversion, conformance testing, mobile communications, LANs

15. NUMBER OF PAGES

150 pages

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

1 Foreword

The final report summarizes research results under ARO/CECOM contract No. DAAL03-92-G-0184. The research covered five areas in protocol engineering for multimedia communications: protocol conversion, conformance testing, multimedia protocol design, LAN extension, and mobile communication. A total of 31 publications have been credited to the contract, including 5 Ph.D. dissertations, 7 journal publications and 19 publications in the proceedings of international conferences. Included in the appendices are five reprints of publications that were not included in previous technical reports.

The research group should like to express their appreciations to Drs. William A Sander (ARO) and Charles J. Graff (CECOM) for their support and leadership in promoting research on protocol engineering.

Contents

4. Appendices

A. C. J. Wang and M. T. Liu, "Automatic Test Case Generation for Estelle," *Proc. of 1993 International Conference on Network Protocols*, pp. 774–781, October 1993.

B. H. Yoon, M. T. Liu, K. Y. Lee and Y. M. Kim, "The Knockout Switch Under Nonuniform Traffic," *IEEE Trans. on Communications*, vol. 43, pp. 2149–2156, June 1995.

C. C. H. Wu, L. S. Koh, and M. T. Liu, "A Synchronization and Compensation Protocol for Multimedia Communication Systems," *Proc. of 1995 International Conference on Network Protocols*, pp. 252–259, November 1995.

D. Y. Yang, T. H. Lai, and M. T. Liu, "Mobile Real-Time Communications in FDDI Networks," *Proc. of 1995 International Conference on Network Protocols*, pp. 201–208, November 1995.

E. M. T. Liu, "Network Interconnection and Protocol Conversion," accepted; to appear in *Advances in Computers*, vol. 40, Academic Press, January 1996.

2 Report Body

In this section, the research results are presented. The research covered five areas: protocol conversion, conformance testing, multimedia protocols design, LANs extension and mobile communication. The problems that were investigated in the respective areas are first described in Section 2.1, then the major results of the research are summarized in Section 2.2. A list of publication credited to this grant and the participating personnel can be found in Section 2.3 and Section 2.4, respectively.

2.1 Statement of Problems

2.1.1 Protocol Conversion

During the past decades, many different network architectures and communication protocols have been successfully developed and widely used. Even though this diversity benefits users with different computational needs, it creates problems when two systems residing in two different networks must communicate with each other. To provide interoperability for two systems that employ different protocols, protocol converters are needed to ensure that the transition sequences of one protocol are correctly converted to the transition sequences of the other protocol. Although most of the existing protocol converters are constructed in ad hoc manner [32, 33, 34, 35], formal methods are needed to obtain solutions that are simpler and more reliable in the face of more and more complex protocols. Recently, some formal methods have been proposed; however, these methods always needs a lot of human ingenuity and the resulted converters are always far from satisfaction. Furthermore, additional steps are needed to verify the resulted converters for their completeness and correctness. The objective of the research is to develop formal methods for constructing protocol converters so that the aforementioned problems can be overcome.

2.1.2 Conformance Testing

Conformance testing is needed to make sure that an implementation of a protocol specification conforms to its specification. To conduct conformance testing, a set of test sequences that can effectively test the implementation must be generated from the specification. Although there are many formal methods to produce test sequences for protocols described in the Finite State Machine (FSM), there is still lack of good methods to do the task for protocols specified in the Extended Finite State Machine (EFSM). The EFSM is a more adequate model than the FSM for specifying multimedia protocols. For protocols specified in EFSM, both the control aspects as in FSM and the data aspects are needed to be tested. As a result, test sequence generation methods for FSM are not adequate for protocols specified in EFSM. Our research objective is to develop test sequence generations for protocols specified in EFSM.

2.1.3 Multimedia Protocols Design

To make multimedia applications a reality in the near future, two important problems need to be solved first: how to provide timely transmission of multimedia data over networks, specifically, local area networks and how to maintain synchronization between media data while playing back. Multimedia data is different from the traditional text data in three ways. First, multimedia data integrate video, voice, image and text together. Each media has its own characteristics and QoS requirement to be fulfilled. Second, there are temporal relationships between different media. These relationships need to be maintained during presentation. Finally, multimedia data may have stringent delay requirement. Traditional local area networks are not quite suitable for transmission of multimedia data because they are designed to handle regular data which are bursty in nature but allow variable delay. If multimedia data are transmitted in local area networks using existing protocols, the time-critical data must compete with regular data and will suffer intolerable delay. Therefore, our first objective is to propose schemes in LAN to ensure timely transmission of the multimedia data. In addition, even if timely transmission of multimedia data can be provided, there is another problem of media synchronization. Since different media have different characteristics and requirements, they are often transmitted through networks in several media streams which may introduce different network jitter and data loss during transmission. As a result, media data may not be synchronized with each other during presentation. Our second objective is to propose synchronization and compensation protocols to ensure the temporal relationships between media are maintained at the receiving sites while playing back.

2.1.4 LANs Extension

The transfer rates required by today's demanding applications are beyond the typical bandwidth of traditional LANs. Video traffic, for example, requires large bandwidth (i.e. 25 Mbps) compared to regular data traffic. A LAN with 10 Mbps bandwidth is unlikely to satisfy the needs of such applications. Although highspeed networks, such as FDDI, DQDB and ATM, have been proposed to replace traditional LANs, the replacement may take decades to finish. Moreover, those traditional LANs such as Ethernet are widely used nowadays. Concerning the replacement cost, availability, and network utilization, the traditional LANs in some places may not be replaceable at all. Therefore, there is a need to extend the current LANs to support new kind of services and applications. Our goal in this part of research is to propose schemes to extend and improve the current LANs to provide services for the multimedia applications.

2.1.5 Mobile Communication

With the introduction of a wide variety of portable computing devices in recent years, the needs to support mobile users with ability to access the existing services provided on the wired networks have attracted a lot of research interests. In order to provide seamless access for these users, two important issues must be addressed. The first issue is to extend the existing wired networks with the ability to cope with users' mobility. Given FDDI's popularity as an underlying network for real-time computing, our objective is to investigate the issue of supporting real-time communications in an FDDI-based mobile network. Note that real-time communication is more complicated than other services available in FDDI networks. If we can extend this service with mobility, providing other services to mobile users will be relatively easy. Supporting real-time communications in regular (non-mobile) FDDI networks has been studied in [36, 37, 38, 39]; however, new technical issues that do not exist before will arise in the new environment. The second issue is the design of wireless multiple access control (MAC) protocols. Since the radio channel bandwidth is limited and the demand for access will surge, the design of an appropriate MAC protocol is crucial for effectively carrying various services between mobile users and wireless interfaces in the wired networks. Our goal in this issue is to design a MAC protocol for voice/data integration.

2.2 Summary of Results

2.2.1 Protocol Conversion

Our major contribution to the protocol conversion problem is to propose the Synchronizing Transition Set approach [9, 4]. This solution is motivated by our observation that the main problems of existing methods are, indeed, due to the fact that the information of the protocol implementation was not taken into consideration during the converter construction process. For a given service specification, there may be a number of different implementations for a protocol. Since the goal of protocol conversion is to find a correct converter for an existing protocol implementation, the information on the existing protocol specification must be taken into consideration for a correct conversion. Therefore, based on the protocol specifications and the service specifications of two targeted protocols, and a service requirement describing services that must be converted between these protocols, our algorithm will first derive the conversion service specification, and then compose the final converter using the information on the existing protocols implementation at a later step. The converter so constructed has three most desirable properties of a correct protocol converter; namely, conformity property, liveness property, and transparency property. The main advantage of the approach is that, while other existing methods require an extra step to verify that some subsets of the required services are supported by a final converter, our method can guarantee that a final converter is complete without any further verification. In addition, our algorithm can handle conversion between sequences of transitions. This capability is crucial when dealing with multimedia protocols in which the semantics of sequences of transitions is different from that of a simple concatenation of the semantics of each original individual transition and the conversion can only be done in a unit of sequence of transitions. The algorithm is initially developed for protocols specified in FSM, and it is later extended to work for protocols specified in EFSM.

In addition, for performing protocol conversion on multimedia networks [9], a more efficient conversion model, called the compensation model, is proposed to replace the traditional intermediate converter model. Under this protocol compensation model, the Synchronizing Transition Set algorithm is extended and modified to accommodate the high speed and high connectivity of the multimedia networks. The modified algorithm still retains all the desirable strengths of the original algorithm.

2.2.2 Conformance Testing

Our main contributions to the problem are that we propose an axiomatic approach for generating test sequences for protocols specified in EFSM [3, 19, 23] and algorithms for generating test sequences for different types of faults [14, 16, 24].

The axiomatic approach is based on the fact that, to construct test sequences for EFSM, one must have a way to trace the changes and dependencies of the data items. So, we design axioms to track how the status of a data item is changed before and after a statement so that after a program is evaluated, the resulting assertion contains a test sequence. Based on this idea, a test procedure is proposed [3]. An improved version that simplifies the axioms, assertions, and the algorithms is also proposed [20] so that the axiomatic approach can be easily applied to any statement set. To extend the power of the approach, a method that takes in a requirement as a guide to generate test sequences for checking requirement conformity is proposed [19]. Most of the test sequence generation methods check transfer errors which are syntactic in nature. By using the requirements, semantic can be introduced into conformity testing. This enables test sequences to be generated with meaningful test purposes.

In addition, since methods of generating test sequences for testing EFSM focus mainly on the data flow aspect of a specification, the control aspect of the specification is ignored. In order to enhance the power of test sequences for EFSM, a method, called effective domain for testing, is proposed [23] to introduce UIO check sequences [40] into the testing of the data flow of a protocol specification modeled by EFSM. By using the concept of effective domain, one can reduce the overall length of test sequences needed for checking both control flow and data flow.

The second contribution is to develop a more flexible approach to generate test sequences for a given fault model [14]. Currently, most of the test generation procedures appearing in the literature generate test sequences to detect a specific type of errors, called a fault model. Since a test method is designed for a fixed fault model, the ability of detecting errors in the implementation is limited. To detect other types of faults, procedures that can generate test sequences based on a given fault model are developed [24]. This method has also been extended [16] to generate test sequences for protocol specification written in the Estelle [41] which is an ISO standard specification language.

In addition, to take advantage of the availability of the existing validation software, a method is also proposed [24] to transform the problem of test sequences generation based on a fault model to a protocol validation problem. Protocol validation is used to determine whether a protocol specification is correct. This area has been studied for years, and many validation tools can be used to generate test sequences. By doing so, we can use these validation tools for the purpose of test sequences generation.

2.2.3 Multimedia Protocols Design

Our contributions include two parts. First two priority schemes are introduced into local area networks [21]. They are the bus-time multiplexing priority scheme and the station multiplexing scheme. Second, multimedia synchronization and compensation protocols are proposed [20, 27] to cope with different delay jitters and loss rates in several media streams. These protocols use a marker, buffering and scheduling techniques to ensure quality multimedia presentation.

To provide timely delivery of multimedia data in traditional local area networks, two level of priorities, high priority (for multimedia or real-time traffic) and low priority (for regular data), are introduced for data transmission [21]. Since traditional local area networks do not support any priority scheme, we propose two priority schemes for implementation in them. The basic idea behind these schemes is to let the stations wishing to transmit multimedia data inform other stations to withhold from transmitting any regular data. Simulation results show that by incorporating these priority schemes into local area networks, response time for multimedia traffic is improved and packets discarding possibilities for multimedia data packets are reduced.

We propose two synchronization protocols to ensure that the quality of service is maintained at the receiver site. The first synchronization protocol we proposed [20] serves as an interface between the underlying ATM network and the multimedia applications. It consists of two entities: a marker and a synchronizer. The marker inserts some control cells, called sync cells, into the media streams to mark the places where data should be synchronized. The synchronizer recognizes the control cells and align the data delivery in different media streams. To cope with the loss of sync cells, three policies, drop-old, transmit-old and delayed-transmit, are considered for implementation within the synchronization protocol. For these three policies, the translation from the quality of service (QOS) specified by the multimedia applications into the QOS for the ATM network has been analyzed. According to the analysis, it can be deduced that the drop-old policy is more demanding on network cell loss rates. However, the drop-old policy is less restrictive on network delay requirement. The delayed-transmit policy imposes stronger demand on network delay, but it allows a smaller portion of cells to be received correctly in that delay period. In addition, it requires a smaller buffer size.

The second proposed protocol [27] combines the scheduling scheme and buffering scheme with underflow threshold to solve the synchronization and compensation problems at the same time. This protocol requires a synchronizer close to the destination. In this protocol, the multimedia data are synchronized at the synchronizer by sending out data according to the transmission schedules at the synchronizer. The synchronization then is maintained at the destinations by minimizing the buffer sizes of the destination devices. Besides, the schedule of a synchronizer is proposed to be modified to cope with the possible starvation problem at destination devices. The underflow threshold is also used in the synchronizer's buffers to prevent the violation of QOS requirements. By controlling the buffer sizes and modifying transmission schedule of the synchronizer, synchronization can be achieved without synchronizing clocks during a connection in this protocol. This is especially suitable to an environment that has limited bandwidths, such as wireless communications. A simulation is performed for comparison of four synchronization schemes: scheduling schemes (with and without clock synchronization), marker scheme, and the proposed protocol. The results confirm that the proposed protocol guarantees satisfaction of QOS requirements and also performs better than the other schemes in terms of media synchronization and freedom from starvation at destination devices.

2.2.4 LANs Extension

In this part of research, our main contribution is to propose the use of switch hubs in LAN configuration in order to increase the effective bandwidth. We study various LAN configurations and their suitability for today's applications. We found that the use of switching hubs in LAN configurations increases the effective bandwidth by supporting multiple concurrent communications [26]. To further improve the performance as well as increasing the reliability and minimizing the bottlenecks, a configuration with multiple distributed switching hubs should be used. We evaluate the performance and reliability of various hub configurations. Our simulation results indicate that the use of hub-network configuration in a local area network enhances its performance. We also study the performance of different switch architectures using simulation and analytical techniques [29]. It shows that parallel switches work very well with high traffic load and, at light-to-medium load, they give close to optimal performance. They provide much better overall performance than traditional switching. Moreover, the effect of the buffer size within the switch is also studied [25]. The result shows that packet delay decreases as the buffer size increases. The reduction is significant when the buffer size is very small and then it slows down as the buffer size grows larger. In addition, higher loads show higher sensitivity in packet delay as the buffer size changes. The percentage of delayed packets is higher than the percentage of dropped packets for the same offered traffic load.

2.2.5 Mobile Communication

Our main contributions include two parts. On the extension of wired networks to support mobility, we propose FDDI-based mobile networks architecture, identify various problems that arise due to mobile host's mobility, and propose solution to minimize the overhead bandwidth to cope with mobility [28, 12]. On the issue of wireless MAC protocol, we propose a mini-packet reservation multiple access control (MPRMA) protocol to support voice/data integration [30, 31].

The proposed structure of an FDDI-based mobile network is depicted in Figure 1. It includes a *control station* (CS), *base stations* (BS), *mobile hosts* (MH), and possibly other stations, which are regular FDDI stations having nothing to do with wireless infrastructures. Stations are connected by an FDDI network; MHs communicate with BSs via wireless media. The CS is a specific station that manages the setups and tear-downs of mobile connections. An MH is a host (or end system in the ISO terminology) that can move around while retaining network connection without disruption. A BS, equipped with radio devices, can communicate with MHs via radio. It serves as a gateway between MHs and the backbone network. Each BS is associated with a *cell*, which is the geographical area covered by its wireless transmission. A BS can communicate with the MHs within its cell, but not those outside the cell. When an MH moves from one cell to another, a *handoff* is said to have occurred.

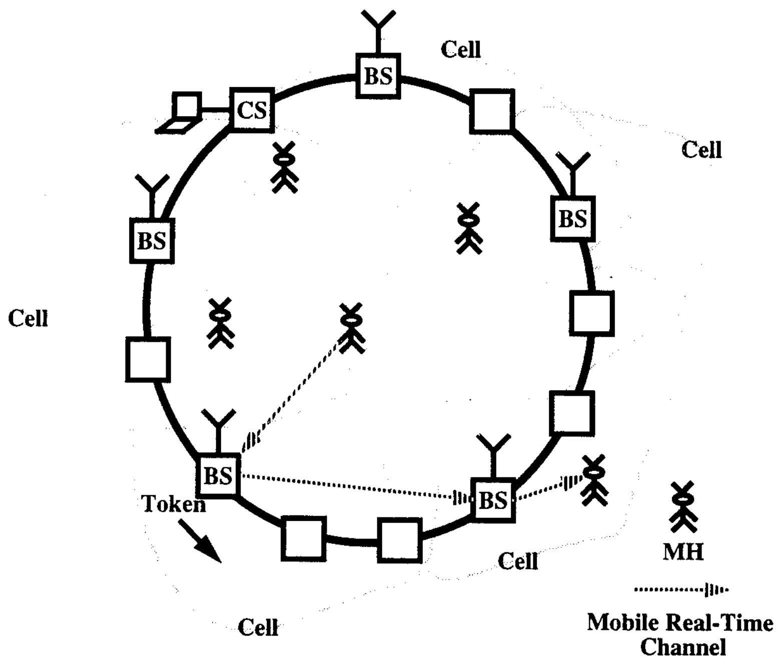


Figure 1: An architecture for FDDI-based mobile networks

In real-time communications, a message sent from an MH to another MH must be delivered in a timely fashion from the source MH to a BS, from the BS to another BS, then from the later BS to the destination MH. In this new mobile environment, due to the handoffs of MHs, some new technical issues that do not exist before arise. The problems that we have identified include the delay problem, the ordering problem, the overhead bandwidth problem, and the bandwidth management problem. To cope with these problems, we propose a destination handoff protocol, a source handoff protocol, an overhead bandwidth management scheme, and an approximate algorithm that attempts to minimize the handoff overhead bandwidth. All these are designed to guarantee mobile real-time channels' QOS during handoffs with as little overhead bandwidth as possible. The proposed protocols are compatible to the current FDDI standards in that we can add base stations that implement our protocols to an existing FDDI network without making any modification to the non-base stations which were originally in the network.

To more efectively support voice/data integration over the wireless link, we propose the MPRMA protocol that combines random access with time division multiple access (TDMA). Instead of using information packets to contend for reservation, the protocol uses smaller reservation packets for contention. Thus, regular TDMA slots can be further divided into smaller slots for sending reservation packets. As a result, collisions caused by contentions can be significantly reduced. Since the roundtrip propagation delay is almost negligible in the microcellular environments, this additional roundtrip for securing a regular slot will not cause much delay to a request. Furthermore, in order to facilitate subsequent packet transmission, a continuation flag is used in each information packet for requesting further reservation. Comparing to the Packet Reservation Multiple Access (PRMA) [42, 43], it is shown by simulation that MPRMA can support more voice and data users, respectively, in a single traffic type environment. And more importantly, MPRMA is also a better alternative to PRMA for supporting voice-data integration. In order to counter adverse channel conditions, a strategy is also proposed. In addition, MPRMA requires no deadline scheduling at the base stations and minimizes overhead bandwidth of the downlink for sending acknowledgment messages.

2.3 List of Publications

The following is a list of 31 publications that have been credited to the grant.

2.3.1 Ph. D. Dissertations (5)

[1] I. M. Hsu, *Distributed Rule Monitoring in Distributed Active Databases.* PhD thesis, Dept. of Computer and Information Science, The Ohio State University, March 1993.

[2] S. S. Yu, *Test Sequence Generation Methods for Communication Protocols.* PhD thesis, Dept. of Computer and Information Science, The Ohio State University, August 1993.

[3] C. J. Wang, *Automatic Test Case Generation of Conformance Testing for Communication Protocols Specified in Extended Models.* PhD thesis, Dept. of Computer and Information Science, The Ohio State University, June 1994.

[4] H.-W. Jeng, *Service Specification Approach to Protocol Conversion.* PhD thesis, Dept. of Computer and Information Science, The Ohio State University, June 1995.

[5] A. Elsaadany, *High-Speed LAN Performance Analysis and Switch Design.* PhD thesis, Dept. of Computer and Information Science, The Ohio State University, June 1995.

2.3.2 Journal Publications (7)

[6] W. S. Chen, K. Y. Lee, and M. T. Liu, "μ-Star: A Indirect Star Network," *Int'l Journal of Networks*, pp. 261–270, July 1993.

[7] I. M. Hsu, M. Singhal, and M. T. Liu, "Distributed Rule Monitoring in Active Databases," *Integrated Computer-Aided Engineering*, vol. 1, no. 4, pp. 295–309, 1994.

[8] H. Yoon, M. T. Liu, and Y. M. Kim, "The Knockout Switch Under Nonuniform Traffic," *IEEE Trans. on Communications*, vol. 43, pp. 2149–2156, June 1995.

[9] H. W. Jeng and M. T. Liu, "Protocol Conversion in Multiumedia Networks: Simulation and Algorithm," *Simulation*, vol. 64, pp. 51–60, January 1995.

[10] M. T. Liu, "Network Interconnection and Protocol Conversion," acccepted; to appear in *Advances in Computers*, vol. 40, Academic Press, Jan. 1996.

[11] H.-W. Jeng and M. T. Liu, "Protocol Converter Generation Using the STS Approach," accepted, to appear in *Special Issue on Protocol Engineering, Computer Communications*, 1996.

[12] T. H. Lai, Y. Yang, and M. T. Liu, "Real-Time Communications in FDDI-Based Mobile Networks," submitted to *Wireless Networks.*

2.3.3 Publications in Conference Proceedings (19)

[13] S. S. Yu and M. T. Liu, "Utilizing Multiple UIO Sequences and Segment Overlap to Shorten Test Sequences," *Proc. ICCS/ISITA'92*, pp. 1291–1296, November 1992.

[14] C.-J. Wang and M. T. Liu, "Generating Test Cases for EFSM with Given Fault Models," in *Proc. IEEE INFOCOM '93*, pp. 774–781, March 1993.

[15] C. J. Wang, P. Krueger, and M. T. Liu, "Intelligent Job Selection for Distributed Scheduling," in *Proc. 13th International Conference on Distributed Computing Systems*, pp. 517–524, June 1993.

[16] C. J. Wang and M. T. Liu, "Automatic Test Case Generation for Estelle," in *Proc. of 1993 International Conference on Network Protocols*, pp. 774–781, October 1993.

[17] S. S. Yu and M. T. Liu, "The Transition-State Pair Method for Test Sequence Generation," in *Proc. IEEE Globecom'93*, pp. 1029–1033, November 1993.

[18] I. M. Hsu, M. Singhal, and M. T. Liu, "Performance Study of Distributed Rule Evaluation Algorithm in Distributed Active Databases," in *Proc. 13th IEEE Annual Int'l Phoenix Conf. on Computers and Communications*, pp. 24–30, April 1994.

[19] L. S. Koh, C. J. Wang, and M. T. Liu, "A Functional Model for Test Sequence Generation," in *Proc. 13th IEEE Annual Int'l Phoenix Conf. on Computers and Communications*, pp. 336–342, April 1994.

[20] C. J. Wang, L. S. Koh, C. H. Wu, and M. T. Liu, "A Multimedia Synchronization Protocol for ATM Networks," in *Proc. 14th International Conference on Distributed Computing Systems*, pp. 476–483, June 1994.

[21] A. Elsaadany, M. Singhal, and M. T. Liu, "Priority Communication Schemes on Local Area Networks for Multimedia Traffic," in *Proc. 19th IEEE Conf. on Local Computer Networks*, pp. 372–379, October 1994.

[22] M. T. Liu, H.-W. Jeng, and L. S. Koh, "Formal Description Techniques for Protocol Specification," in *Proc. of ATR Int'l Workshop on Communications Software Engineering*, pp. 31–71, October 1994.

[23] L. S. Koh and M. T. Liu, "Test Path Selection Based on Effective Domains," in *Proc. 1994 IEEE Int'l Conf. on Network Protocols*, pp. 64–71, October 1994.

[24] C. J. Wang, L. S. Koh, and M. T. Liu, "Protocol Validation Tools as Test Case Generators," in *Proc. 7th IFIP Int'l Workshop on Protocol Test Systems*, pp. 149–164, November 1994.

[25] A. Elsaadany, M. Singhal, and M. T. Liu, "Performance Study of Buffering within Switches in Local Area Networks," in *Proc. of 4th International Conference on Computer Communications and Networks*, September 1995.

[26] A. Elsaadany, M. Singhal, and M. T. Liu, "Alternative Configurations for Local Network Design," in *Proc. 20th IEEE Conf. on Local Computer Networks*, October 1995.

[27] C. H. Wu, L. S. Koh, and M. T. Liu, "A Synchronization and Comensation Protocol for Multimedia Communication Systems," in *Proc. of 1995 International Conference on Network Protocols*, pp. 252–259, November 1995.

[28] Y. Yang, T. H. Lai, and M. T. Liu, "Mobile Real-Time Communications in FDDI Networks," in *Proc. of 1995 International Conference on Network Protocols*, pp. 201–208, November 1995.

[29] A. Elsaadany, M. Singhal, and M. T. Liu, "Performance Evaluation of Switching in Local Area Networks," submitted to *IPCCC'96*, March 1996.

[30] L. S. Koh, C. H. Wu, and M. T. Liu, "Dynamic Contention Strategies for PRMA," submitted to *ICDCS'96*, May 1996.

[31] L. S. Koh and M. T. Liu, "A Wireless Multiple Access Control Protocol for Voice-Data Integration," submitted to *ICPADS'96*, June 1996.

2.4 Participating Personnel

In addition to Principal Investigator, Dr Ming T. Liu, the following 8 Graduate Research Associates have participated in the research project:

- Ms. Ing-Miin Hsu (Ph. D., March 1993)
- Ms. Sarah S. Yu (Ph. D., March 1994)
- Chang-Jia Wang (Ph. D., June 1994)
- Hou-Wa Jeng (Ph. D., June 1995)
- Amr Elsaadany (Ph. D., June 1995)
- Liang-Seng Koh (Ph. D., June 1996, expected)
- Yibin Yang (Ph. D., August 1996, expected)
- Ms. Chao-Hui Wu (Ph. D., December 1996, expected)

3 Bibliography

[32] P. Francois and A. Potocki, "Some Methods for Providing OSI Transport in SNA," *IBM Journal of Research & Development*, vol. 27, pp. 452–463, Sept. 1983.

[33] I. Groenback, "Conversion Between the TCP and ISO Transport Protocols as a Method of Achieving Interoperability Between Data Communication Systems," *IEEE JSAC*, vol. SAC-4, pp. 288–296, Mar. 1986.

[34] J. M. Rodriguez, "An X.PC/TCP Protocol Translator," in *Proc. IEEE INFOCOM 1988*, pp. 308–313, Mar. 1988.

[35] M. T. Rose and D. E. Cass, "OSI Transport Service on Top of TCP," *Computer Networks and ISDN Systems*, vol. 12, pp. 159–173, 1987.

[36] G. Agrawal, B. Chen, and W. Zhao, "Local Synchronous Capacity Allocation Schemes for Guranteeing Message Deadlines with the Timed Token Protocol," in *Proc. INFOCOM*, March 1993.

[37] G. Agrawal, B. Chen, W. Zhao, and S. Davari, "Guranteeing Synchronous Message Deadlines with the Timed Token Protocol," in *Proc. IEEE International Conference on Distributed Computing Systems*, June 1992.

[38] B. Chen, G. Agrawal, and W. Zhao, "Optimal Synchronous Capacity Allocation for Hard Real Time Communications with the Timed Token Protocol," in *Proc. Real Time Systems Symposium*, December 1992.

[39] Q. Zheng and K. G. Shin, "On the Ability of Estabishing Real-Time Channels in Point-to-Point Packet-Switched Networks," *IEEE Transactions on Communications*, vol. COM-42, no. 2/3/4, pp. 1096–1105, 1994.

[40] K. Sabnani and A. Dahbura, "A Protocol Test Generation Procedure," *Computer Networks and ISDN Systems*, vol. 15, pp. 285–297, 1988.

[41] S. Budkowski and P. Dembinski, "An Introduction to Estelle: A Specification Language for Distributed Systems," *Computer Networks and ISDN Systems*, vol. 14, no. 1, pp. 3–23, 1987.

[42] D. J. Goodman, R. A. Valenzuela, K. T. Gayliard, and B. Ramamurthi, "Packet reservation multiple access for local wireless communications," *IEEE Trans. on Communications*, vol. 37, pp. 885–890, August 1989.

[43] D. J. Goodman, "Cellular packet communications," *IEEE Trans. on Communications*, vol. 38, pp. 1272–1280, August 1990.

A. **C. J. Wang and M. T. Liu,**
"Automatic Test Case Generation for Estelle,"
Proc. 1993 IEEE Int'l Conf. on Network Protocols,
pp. 774-781, October 1993.

Automatic Test Case Generation for Estelle*

Chang-Jia Wang and Ming T. Liu

Department of Computer and Information Science
The Ohio State University
Columbus, OH 43210-1277, USA

Abstract

In this paper, an automatic test case generation method for Estelle is proposed. A formal model is introduced to describe the dynamic properties of Estelle specifications so as to verify the difference between the behavior of the specification and the behavior of its fault models. Based on the difference, an algorithm is presented to produce test cases that can detect such implementation faults. The algorithm can generate test cases not only for single module specifications, but also for systems containing multiple modules that run concurrently. In addition, heuristics are suggested to improve the performance of the test case generation process.

1 Introduction

Protocol conformance testing is a major issue in the design of reliable communication networks. It ensures that a *protocol implementation* is consistent with its *specification* in various hardware and software environments. One major issue of conformance testing is *test case generation*. Most of the test case generation methods proposed are based on the *Finite State Machine (FSM)* model [1, 2, 3, 4]. However, the FSM model can only specify problems within the domain of regular languages. To solve more general problems, other models such as the *Extended Finite State Machine (EFSM)* [5], *Estelle* [6] or *LOTOS* [7] are used [8, 9, 10, 11, 12, 13, 14]. These Models usually contain memories. Therefore, in addition to testing control flow, the data aspect of the models needs to be considered as well.

A test case is meaningless if one does not know its purpose. A *test purposes* is a set of statements showing what type of error a test case tries to detect. The error type is called a *fault model* [15], which must be given before any meaningful test case can be generated. The test methods mentioned above are used to generate test cases for a fixed fault-model. When one needs the confidence that certain critical faults will not occur, these methods cannot guarantee the detection of the cirtical faults if they are not included in the fixed fault-models.

It is infeasible to generate a test case for every possible fault. Thus, it is desirable to provide the protocol designer with a freedom of choosing whatever faults he/she believes are important. Based on our previous work [13, 14], a test case generation method for EFSM that produces test cases for given fault models [16] has been developed. The method employs a program verification technique, called axiomatic semantics [17, 18, 19], to symbolically verify a protocol specification. The difference between the behavior of the specification and the behavior of the fault model is analyzed and then test cases are generated based on the difference.

Since Estelle has been adopted as an international standard specification language for communication protocols, it is interesting to extend the proposed method in [16] from EFSM to Estelle. In this paper, an automatic test case generation method for Estelle is proposed. After axioms for Estelle statements are defined, the same algorithm in [16] can be used to generate test cases for given fault models. In addition, the same method is extended to generate test cases for multiple modules running concurrently. Furthermore, a heuristic search method is also suggested to improve the performance of the search algorithm.

The rest of this paper is organized as follows: In Section 2, the behavior model proposed in [16] is briefly introduced. Then, axioms for Estelle statements are defined in Section 3, and the definition of a test case and how to generate it are presented in Section 4. Finally, conclusions are given in Section 5.

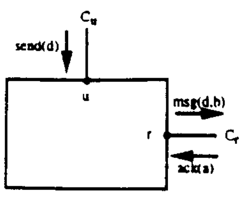
2 Background

2.1 Estelle and EFSM

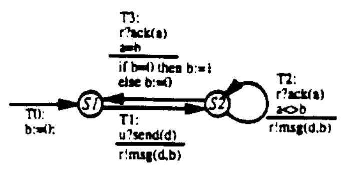
Estelle [6] is a description language for *Extended Finite State Machines (EFSM)*. An EFSM is a *Finite State Machine (FSM)* with memory (called *variables* hereafter). Similar to an FSM, an EFSM contains a set of *states* and *transitions* pointing from one state (called *head state*) to another (called *tail state*). There are usually a *condition* and an *action* associated with each transition. The action of a transition can be executed only when the EFSM is at

*Research reported herein was supported by U.S. Army Research Office, under contracts No. DAAL03-91-G-0093 and No. DAAL03-92-G-0184. The views, opinions, and/or findings contained in this paper are those of the authors and should not be construed as an official Department of the Army position, policy or decision.

0-8186-3670-X/93 $03.00 © 1993 IEEE



(a)



(b)

```
initialize                    from S1
to S1                         to S2
name T0                       when u.send(d)
    begin                     name T1
    b:=0                          begin
    end;                          output r.msg(d,b)
                                  end;

from S2                       from S2
to S2                         to S1
when r.ack(a)                 when r.ack(a)
provided ab                   provided a=b
name T2                       name T3
    begin                         begin
    output r.msg(d,b)             if b=0 then b:=1
    end;                          else b:=0
                                  end;
```

(c)

Figure 1: The sender of an ABP protocol

the head state of the transition and the condition is satisfied. After the action is executed, the EFSM moves to the tail state of the transition and becomes ready for executing next transition. In Estelle, the head and tail states are defined in **from** and **to** clauses, respectively. The condition of a transition includes **when** and **provided** clauses. The former inputs a message from an *interaction point (IP)* and the latter imposes a restriction under which the action can be executed. The action is a Pascal statement sequence quoted between reserved words **begin** and **end**. The statements can be executed only when both the input in **when** clause is available and the condition in **provided** clause is satisfied. Readers who are not familiar with the Estelle syntax are referred to [6].

In addition to the clauses described above, there are declaration statements and other clauses that are not important for generating test cases. Therefore, in the rest of this paper, we focus only on the portion of specification that defines the transitions, and assume that all modules, channels, IPs, and variables are properly defined.

A sample Estelle specification is shown in Figure 1, which is the sending side of an *Alternating Bit Protocol (ABP)*. The structure of the protocol is shown in Figure 1a, the state diagram is shown in Figure 1b, and corresponding Estelle transitions are defined in Figure 1c.

2.2 Behavior of an EFSM

Estelle is able to clearly describe the structure of an EFSM. However, knowing the structure alone is inadequate for generating test cases. In order to find test cases for an EFSM, one needs to know the *dynamic property* of the EFSM as well. Specifically, the knowledge of how the variables in the EFSM change their values is needed. Therefore, the *behavior model* proposed in [16] is used in this paper to describe such changes.

When a transition of an EFSM is executed, the values of the variables change accordingly. The values are preserved by their *versions*, which provide names to store different values of the same variable during different periods of time. The versions are denoted by an operator ν (pronounced "new") followed by the names of the variables. For example, νx denotes a newer version of variable x. Similarly, $\nu\nu x$ (or simply $\nu^2 x$) denotes a newer version of νx, and $\nu^0 x$ (or simply x) denotes the original version of variable x.

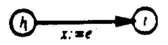
The relationship between versions and their values is called a *scenario*. Formally, a scenario is a function that maps a set of versions into a set of values. It is denoted as a sequence of versions and their corresponding values quoted by a pair of angle brackets ($\langle\,\rangle$). For example, a scenario, $\langle x = 0, y = 0, \nu x = 1, \nu y = 0, \ldots\rangle$, shows that x and y are originally zeros, and then change to 1 and 0 later, respectively.

An *assertion* is a set of scenarios. It is denoted by a boolean expression, called *predicate*, quoted by a pair of braces ($\{\,\}$). The assertion contains those scenarios that satisfy the predicate. For example, assertion $\{\nu y \geq 0\}$ contains both scenarios $\langle \nu x = 1, \nu y = 0, \ldots\rangle$ and $\langle \nu x = 1, \nu y = 1, \ldots\rangle$. The set of all possible scenarios that can be developed after a state, say a, is denoted by $\beta[a]$. For instance, $\beta[a] = \{\nu y \geq 0\}$ means that if the EFSM is at state a, every possible scenario developed thereafter has the property that the first version of y is greater than or equal to zero.

Let M be an EFSM. The behavior of M is denoted as $\beta[M]$, which is defined as $\beta[M] = \bigcup_{\forall s} \beta[s]$, where s is a state of M. In addition, let A_1 and A_2 be two assertions, and p_1 and p_2 be two predicates such that $A_1 = \{p_1\}$ and $A_2 = \{p_2\}$. It can be shown that $A_1 \cup A_2 = \{p_1 \vee p_2\}$, $A_1 \cap A_2 = \{p_1 \wedge p_2\}$, $\overline{A_1} = \{\neg p_1\}$, and $A_1 - A_2 = \{p_1 \wedge \neg p_2\}$.

2.3 Axioms

A rule that describes how a statement changes the course of the scenarios' development is called an *axiom*. Figure 2 is a simple example for the axiom of an assignment statement. The transition starts from state h, points to state t, and contains only one action: "$x := e$," which assigns the result of expression e to variable x. The axiom means that the scenarios that can be developed after state h is the same as the scenarios developed after state t with one more restriction that the new version of variable x is equal to the result of expression e (denoted by assertion $\{\nu x = e\}$). Since variable x has been upgraded, any development of variable x after state t must start from the upgraded version. Therefore, the second term of the axiom is denoted by $\beta[t]|^{\{\nu x\}}$, meaning that every



$\beta[h] = \{\nu x = e\} \cap \beta[t]|^{\{\nu x\}}$

Figure 2: An example of an axiom

occurrence of x in $\beta[t]$ is replaced by νx. In general, for any assertion A, $A|^{\{\nu^i x\}}$ denotes replacing each occurrence of $\nu^j x$ in A by $\nu^{i+j}x$. For example, if $A = \{x = 0 \wedge \nu x = 1\}$, then $A|^{\{\nu x\}} = \{\nu x = 0 \wedge \nu^2 x = 1\}$.

Let A_T be the assertion that indicates the restriction imposed by T, and let V_T be the set of new versions created by T. If the set of scenarios developed after state h through transition T is denoted as $\beta[h]_T$, then $\beta[h]_T = A_T \cap \beta[t_T]|^{V_T}$, meaning that the scenarios developed after h through T are those developed after t_T (with upgraded versions) that satisfy A_T. The *parallel composition* of the behavior at state h ($\beta[h]$) means that $\beta[h]$ is the combination of all the transitions fan out from h. Thus, $\beta[h]$ can be derived as follows:

$$\begin{aligned} \beta[h] = {} & \beta[h]_{T_1} \cup \beta[h]_{T_2} \cup \cdots \cup \beta[h]_{T_n} \\ = {} & (A_{T_1} \cap \beta[t_{T_1}]|^{V_{T_1}}) \cup (A_{T_2} \cap \beta[t_{T_2}]|^{V_{T_2}}) \cup \cdots \\ & \cup (A_{T_n} \cap \beta[t_{T_n}]|^{V_{T_n}}) \end{aligned}$$

The *serial composition* of $\beta[h]$ means that $\beta[h]$ is the resulting scenario set derived from a sequence of transitions after state h. Therefore,

$$\begin{aligned} \beta[h] = {} & A_{T_1} \cap \beta[t_{T_1}]|^{V_{T_1}} \\ = {} & A_{T_1} \cap (A_{T_2} \cap \beta[t_{T_2}]|^{V_{T_2}})|^{V_{T_1}} \\ = {} & A_{T_1} \cap A_{T_2}|^{V_{T_1}} \cap \beta[t_{T_2}]|^{V_{T_1} \uplus V_{T_2}} \\ & \vdots \\ = {} & A_{T_1} \cap A_{T_2}|^{V_{T_1}} \cap A_{T_3}|^{V_{T_1} \uplus V_{T_2}} \cap \\ & \cdots \cap \beta[t_{T_n}]|^{V_{T_1} \uplus V_{T_2} \uplus \cdots \uplus V_{T_n}} \end{aligned}$$

where the operation "$\uplus$" is defined as follows:

Definition 1 *Let V_1 and V_2 be two sets of versions. Then*

$$\begin{aligned} V_1 \uplus V_2 = {} & \{\nu^{i+j} w \mid \exists \nu^i w \in V_1 \wedge \exists \nu^j w \in V_2\} \cup \\ & \{\nu^i w \mid \exists \nu^i w \in V_1 \wedge \forall \nu^j w \notin V_2\} \cup \\ & \{\nu^j w \mid \forall \nu^i w \notin V_1 \wedge \exists \nu^j w \in V_2\} \\ V^n = {} & \underbrace{V \uplus V \uplus \ldots \uplus V}_{n} \\ = {} & \{\nu^{nk} w \mid \nu^k w \in V\} \end{aligned}$$

3 Axioms for Estelle

Estelle specifications can be written in *normal form specifications*, in which every transition is specified in the following format:

from h
to t
when $p.m(x_1, x_2, \ldots, x_n)$
provided B
begin S **end**;

where h and t, respectively, are the head and tail states of the transition, p is an interaction point, m is a message received from p with arguments x_1 through x_n, and B is a boolean expression indicating the condition under which a sequence of Pascal statements, S, can be executed. A transition can be divided into two simpler transitions: the first transition contains **when** and **provided** clauses, and the second executes S. An auxiliary state u can be placed between the two transitions. Then, the axiom for an Estelle transition can be defined as follows:

Axiom 1

$$\begin{aligned} \beta[h] = {} & \{\nu c_p = m(\nu x_1, \nu x_2, \ldots, \nu x_n)\} \cap \\ & \{(\nu x_1 = \nu\kappa) \wedge (\nu x_2 = \nu^2\kappa) \wedge \cdots \wedge (\nu x_n = \nu^n\kappa)\} \cap \\ & \{\nu\tau = \nu c_p\} \cap \\ & \{B\} \cap \\ & \beta[u]|^{\{\nu c_p, \nu x_1, \ldots, \nu x_n, \nu^n\kappa, \nu\tau\}} \end{aligned}$$

There are five terms in the axiom. The first term shows that the current data received from c_p (denoted νc_p) is message m with arguments νx_1 through νx_n. The values of νx_1 to νx_n are shown in the second term, in which an auxiliary variable κ is used to indicate different input values. The third term is used to arrange the external events in chronicle order, where τ is an auxiliary variable used to record current I/O events. Therefore, "$\nu\tau = \nu c_p$" indicates that the current external event is at channel c_p. The fourth term shows that the condition in B must be satisfied. The last term means that the scenario set developed after the transition contains those scenarios developed after state u with variable versions updated.

The remaining problem is to define the axioms for S in order to obtain $\beta[u]$ in terms of $\beta[t]$. In the following axioms, the notations A_s and V_s represent, respectively, the assertion and the set of new versions developed after executing a statement s or a sequence of statement s.

Axiom 2 (Empty Statements)
If S is empty, $\beta[u] = \beta[t]$.

Axiom 3 (Sequential Statements)
If S is "s; S'," where s is a statement and S' is a sequence of statements,

$$\beta[u] = A_s \cap A_{S'}|^{V_s} \cap \beta[t]|^{(V_s \uplus V_{S'})}.$$

Axiom 4 (Assignment Statements)
If S is "$x := e$," where x is a variable and e is an expression,

$$\beta[u] = \{\nu x = e\} \cap \beta[t]|^{\{\nu x\}}$$

Axiom 5 (Output Statements)
*If S is "**output** $p'.m'(e_1, e_2, \ldots, e_n)$," where p' is an interaction point, m' is an output message, and expressions e_1 to e_n are the arguments of m', then*

$$\beta[u] = \{\nu c_{p'} = m'(e_1, e_2, \ldots, e_n)\} \cap \{\nu\tau = \nu c_{p'}\} \cap \beta[t]|^{\{\nu c_{p'}, \nu\tau\}}$$

where $c_{p'}$ is the corresponding channel of p'.

In Axiom 5, the first term indicates that the next event at channel $c_{p'}$ is $m'(e_1, e_2, \ldots, e_n)$. The second term appends the event to the sequence of external events recorded by τ.

Axiom 6 (Selection Statements)
*If S is "**if** $\mathcal{B}$ **then** S_1 **else** S_2," where $\mathcal{B}$ is a boolean expression and S_1 and S_2 are two sequences of statements,*

$$\beta[u] = (\{\mathcal{B}\} \cap A_{S_1} \cap \beta[t]|^{V_{S_1}}) \cup (\{\neg\mathcal{B}\} \cap A_{S_2} \cap \beta[t]|^{V_{S_2}})$$

A selection statement can be viewed as a parallel composition of two transitions. One can be executed only when condition $\mathcal{B}$ is satisfied, and the other only when it is not.

Axiom 7 (Function Calls)
Let F be a function defined as follows:

```
    Function $F(x_1, x_2, \ldots, x_n)$;
        begin $S_F$; $F := e$ end;
```

where $x_1, x_2, \ldots, x_3$ are parameters of F, S_F is a sequence of statements, and e is an expression. If S is "$y := F(a_1, a_2, \ldots, a_n)$," where $a_1, a_2, \ldots, a_n$ are arguments, then

$$\begin{aligned}\beta[u] \;=\; & \{(\nu x_1 = a_1 \wedge \nu x_2 = a_2 \wedge \ldots \wedge \nu x_n = a_n)\} \cap \\ & A_{S_F}|^{V'} \cap \\ & \{\nu y = e\}|^{V_{S_F} \uplus V'} \cap \\ & \beta[t]|^{V_{S_F} \uplus V' \uplus \{\nu y\}}\end{aligned}$$

where $V' = \{\nu x_1, \nu x_2, \ldots, \nu x_n\}$.

The first term of the axiom indicates that the arguments (the a's) are assigned to the parameters (the x's). The assertion imposed by the body of the function (A_{S_F}) restricts the development of the scenarios after the function call. Therefore, it is included in the second term. The third term shows that after the function call, variable y receives the value from expression e (thus $\{\nu y = e\}$). Finally, the new versions generated by the above are contained in $V_{S_F} \uplus V' \uplus \{\nu y\}$, so the variables in $\beta[t]$ need to be upgraded accordingly. The axiom for procedure calls can be defined in a similar manner.

Axiom 8 (Procedure Calls)
Let P be a procedure defined as follows:

```
Procedure $P(x_1, x_2, \ldots, x_n$, var $y_1$, var $y_2$, ..., var $y_m)$;
    begin $S_P$ end;
```

where $x_1, x_2, \ldots, x_n$ and $y_1, y_2, \ldots, y_m$ are variables and S_P is a sequence of statements. If S is "$P(a_1, a_2, \ldots, a_n, b_1, b_2, \ldots, b_m)$," where $a_1, a_2, \ldots, a_n$ and $b_1, b_2, \ldots, b_m$ are arguments, then

$$\begin{aligned}\beta[u] \;=\; & \{(\nu x_1 = a_1 \wedge \nu x_2 = a_2 \wedge \ldots \wedge \nu x_n = a_n) \wedge \\ & (\nu y_1 = b_1 \wedge \nu y_2 = b_2 \wedge \ldots \wedge \nu y_m = b_m)\} \cap \\ & A_{S_P}|^{V'} \cap \\ & \{\nu b_1 = y_1 \wedge \nu b_2 = y_2 \wedge \ldots \wedge \nu b_n = y_n\}|^{V_{S_P} \uplus V'} \cap \\ & \beta[t]|^{V_{S_P} \uplus V' \uplus V''}\end{aligned}$$

where $V' = \{\nu x_1, \nu x_2, \ldots, \nu x_n, \nu y_1, \nu y_2, \ldots, \nu y_m\}$ and $V'' = \{\nu b_1, \nu b_2, \ldots, \nu b_m\}$.

It is assumed that every variable has its own name. Although this is not true in a real specification, name conflicts can always be resolved by attaching the corresponding function or procedure names to the variable names.

Axiom 9 (While Statement)
*If S is "**while** $\mathcal{B}$ **do** S'," where $\mathcal{B}$ is a boolean expression and S' is a sequence of statements, then*

$$\beta[u] = \bigcap_{k=0}^{n-1} [\{\mathcal{B}\} \cap A_{S'}]|^{V_{S'}^{k}} \cap [\{\neg\mathcal{B}\} \cap \beta[t]]|^{V_{S'}^{n}}$$

Let $\beta[u]_i$ denote the behavior of state u after the i-th iteration, and assume that S' will be executed n times. Then,

$$\begin{aligned}\beta[u] &= (\{\mathcal{B}\} \cap A_{S'}) \cap \beta[u]_1|^{V_{S'}} \\ &= (\{\mathcal{B}\} \cap A_{S'}) \cap (\{\mathcal{B}\} \cap A_{S'})|^{V_{S'}} \cap \beta[u]_2|^{V_{S'}^{2}} \\ &\;\;\vdots \\ &= \textstyle\bigcap_{k=0}^{n-1}(\{\mathcal{B}\} \cap A_{S'})|^{V_{S'}^{k}} \cap \beta[u]_n|^{V_{S'}^{n}} \\ &= \textstyle\bigcap_{k=0}^{n-1}(\{\mathcal{B}\} \cap A_{S'})|^{V_{S'}^{k}} \cap (\{\neg\mathcal{B}\} \cap \beta[t]|^{V_{S'}^{n}})\end{aligned}$$

It is possible to construct a data structure to represent "$\bigcap_{k=0}^{n-1}(\{\mathcal{B}\} \cap A_{S'})|^{V_{S'}^{k}}$" such that n remains undefined; the actual value of n will be determined later when the specification is analyzed. Similarly, **repeat** and **for** statements can be defined as follows.

Axiom 10 (Repeat Statements)
*If S is "**repeat** S' **until** $\mathcal{B}$," where $\mathcal{B}$ is a boolean expression and S' is a sequence of statements, then*

$$\beta[u] = \bigcap_{k=1}^{n} [A_{S'}|^{V_{S'}^{k-1}} \cap \{\neg\mathcal{B}\}|^{V_{S'}^{k}}] \cap [\{\mathcal{B}\} \cap \beta[t]]|^{V_{S'}^{n}}$$

Axiom 11 (For-To Statements)
*If S is "**for** $i := m$ **to** n **do** S'," where i is an integer, m and n are integer expressions, and S' is a sequence of statements, then*

$$\beta[u] = \bigcap_{k=0}^{n-m} [\{\nu i = k + m\} \cap A_{S'}]|^{(\nu i \uplus V_{S'})^{k}} \cap \beta[t]|^{V_{S'}^{n-m+1}}$$

$$\begin{aligned}\beta[S_1] \;=\; & \{(\nu c_u = send(\nu d)) \wedge (\nu d = \nu\kappa) \wedge (\nu\tau = \nu c_u)\} \cap \\ & \{(\nu c_r = msg(\nu d, b)) \wedge (\nu^2\tau = \nu c_r)\} \cap \\ & \beta[S_2] |^{\{\nu c_u, \nu c_r, \nu d, \nu\kappa, \nu^2\tau\}} \\ \beta[S_2] \;=\; & (\{(\nu c_r = ack(\nu a)) \wedge (\nu a = \nu\kappa) \wedge (\nu\tau = \nu c_r)\} \cup \\ & \{(\nu a \neq b) \wedge (\nu^2 c_r = msg(d, b)) \wedge (\nu^2\tau = \nu^2 c_r)\} \cap \\ & \beta[S_2] |^{\{\nu a, \nu^2 c_r, \nu\kappa, \nu^2\tau\}}) \cup \\ & (\{(\nu c_r = ack(\nu a)) \wedge (\nu a = \nu\kappa) \wedge (\nu\tau = \nu c_r)\} \cap \\ & \{(\nu a = b) \wedge ((b = 0 \wedge \nu b = 1) \vee (b = 1 \wedge \nu b = 0))\} \cap \\ & \beta[S_1] |^{\{\nu a, \nu b, \nu c_r, \nu\kappa, \nu\tau\}})\end{aligned}$$

Figure 3: The behavior function of the ABP in Figure 1

Axiom 12 (For-Downto Statements)
*If S is "**for** $i := n$ **downto** m **do** S'," where i is an integer, m and n are integer expressions, and S' is a sequence of statements, then*

$$\beta[u] = \bigcap_{k=0}^{n-m} [\{\nu i = n - k\} \cap A_{S'}] |^{(\nu i \uplus V_{S'})^k} \cap \beta[t] |^{V_{S'}^{n-m+1}}$$

The behavior of the EFSM in Figure 1 can be derived as the equations shown in Figure 3, where $\beta[S_1]$ is derived from Axioms 1 and 5, and $\beta[S_2]$ is derived from Axioms 1, 4, 5, and 6. For simplicity, the initial transition is ignored. Substituting $\beta[S_2]$ into the first equation in Figure 3, $\beta[S_1]$ can be expanded to describe more detailed information about the behavior of the specification at state S_1. Unless every executable path ends up at a final state, the recursive functions never stop. A method to extract a finite external event sequence as a test case will be presented in the next section.

4 Test Case Generation

4.1 Functional Equivalence

Two scenarios are *functionally equivalent* if they generate the same external events. For example, scenarios

$\langle \nu a = 0, \nu d = \nu\kappa, \nu b = 1, \nu c_r = msg(\nu d, \nu b), \nu\tau = \nu c_r \rangle$, and
$\langle \nu a = 1, \nu d = \nu\kappa, \nu b = 1, \nu c_r = msg(\nu d, \nu b), \nu\tau = \nu c_r \rangle$

are functionally equivalent since both output $msg(\nu\kappa, 1)$ to channel c_r as their first external events (because $\nu\tau = \nu c_r = msg(\nu d, \nu b)$, where $\nu d = \nu\kappa$ and $\nu b = 1$). The value of νa is unimportant in the example since it does not affect the outcome. Formally, two scenarios h and g are functionally equivalent if $h(\nu^i\tau) = g(\nu^i\tau)$ for any nonnegative integer i, where $h(\nu^i\tau)$ denotes the value of $\nu^i\tau$ in scenario h.

A *functionally equivalent set (FES)* of an assertion A is a set of all functionally equivalent scenarios of the scenarios in assertion A. The FES of A, denoted **FES**(A), is defined as follows:

Definition 2
$\mathrm{FES}(A) = \{g \mid \forall i \geq 0 . \exists h \in A . [h(\nu^i\tau) = g(\nu^i\tau)]\}$

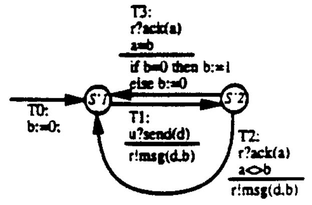


Figure 4: Two mutants for the EFSM in Figure 1

For example, if assertion

$A = \{\nu a = 0 \wedge \nu d = \nu\kappa \wedge \nu b = 1 \wedge \nu c_r = msg(\nu d, \nu b) \wedge \nu\tau = \nu c_r\}$

then

$\mathbf{FES}(A) = \{\nu d = \nu\kappa \wedge \nu b = 1 \wedge \nu c_r = msg(\nu d, \nu b), \nu\tau = \nu c_r\}$

In other words, the **FES**() function removes those predicate terms that are irrelevant to the external events. It can be shown that $\mathbf{FES}(A \cup B) = \mathbf{FES}(A) \cup \mathbf{FES}(B)$ and $\mathbf{FES}(A \cap B) = \mathbf{FES}(A) \cap \mathbf{FES}(B)$.

4.2 Test Cases

A *mutant* is a specification of a fault model. For example, a mutant of Figure 1 is shown in Figure 4, in which the faulty transition T_2 points to a wrong state S_1.

A *test scenario* is a scenario whose external event can distinguish a correct EFSM from its mutants. That is, the external event sequence generated by the test scenario cannot be reproduced by the mutants. A test scenario p that distinguishes a correct EFSM M from a mutant F can be defined as follows:

Definition 3 *A scenario p is a test scenario for mutant F if $p \in (\beta[M] - \mathbf{FES}(\beta[F]))$.*

A *test case* is a prefix of the external event sequence generated by a test scenario. If h is a test scenario for an EFSM M and its mutant F, a test case is defined as follows:

Definition 4 *An event sequence T is a test case if*
1. $T = \langle h(\tau), h(\nu\tau), h(\nu^2\tau), ..., h(\nu^n\tau)\rangle$, and
2. $\forall g \in \mathbf{FES}(\beta[F]), \exists i \leq n, [h(\nu^i\tau) \neq g(\nu^i\tau)]$.

The definition means that, a test case is a finite external event sequence that contains at least one element that cannot be reproduced by a faulty implementation. Therefore, one can recognize an incorrect implementation once an unexpected external event is produced.

4.3 Test Case Generation for Single-Module Specifications

For any EFSM M and its mutant F, it can be proved that $\beta[s] - \mathbf{FES}(\beta[F]) \subseteq \beta[M] - \mathbf{FES}(\beta[F])$, where s is a state of M. Though a test scenario can be found anywhere in $\beta[M]$ – $\mathbf{FES}(\beta[F])$, it is more convenient to select the initial state as state s and to search for a test scenario in $\beta[s]$ – $\mathbf{FES}(\beta[F])$. Finding a test scenario from the latter not only avoids setting up the implementation under test (IUT) to the starting state of the test scenario, but also reveals the correctness of the initial transition.

Let h be a scenario starting from state s. Let u be a state that the corresponding path of h passes through. Then, it can be shown that $h \in A \cap \beta[u]\,|^{V}$, for some assertion A and new version set V. Hence, any set of scenarios, H, can be represented as $\bigcup_h (A_h \cap \beta[u_h]|^{V_h})$, where h is a scenario in H, and A_h, u_h, and V_h are an assertion, a state, and a set of new versions, respectively. Let H and G be two sets of scenarios for EFSM M_h and M_g, respectively, such that $H = \bigcup_h (A_h \cap \beta[u_h]|^{V_h})$ and $G = \bigcup_g (A_g \cap \beta[u_g]\,|^{V_g})$. Then, H – $\mathbf{FES}(G)$ can be derived as follows:

$$H - \mathbf{FES}(G) \supseteq \bigcup_h \bigcap_g K_{h,g}$$

where $K_{h,g}$ =

$$\begin{cases} A_h \cap \beta[u_h]|^{V_h} & \text{if } A_h \cap \mathbf{FES}(A_g) = \emptyset \quad (1a) \\ (A_h - \mathbf{FES}(A_g)) \cap \beta[u_h]|^{V_h} & \text{if } A_h \cap \mathbf{FES}(A_g) \neq \emptyset \wedge \\ & A_h - \mathbf{FES}(A_g) \neq \emptyset \quad (1b) \\ A_h \cap (\beta[u_h]|^{V_h} - \mathbf{FES}(\beta[u_g]|^{V_g})) & \text{otherwise} \quad (2) \end{cases}$$

As shown in the equations above, $K_{h,g}$ can be divided into Cases 1a, 1b, and 2, which are illustrated in Figure 5. Each circle in the figure represents a set of scenarios. In Case 2 (Figure 5a), A_h is covered by $\mathbf{FES}(A_g)$, meaning that for any scenario in A_h, there is a scenario in A_g that generates the same external event sequence as A_h does. Therefore, A_h along cannot distinguish H from G. To find the differences between H and G, $\beta[u_h]\,|^{V_h}$ $-\mathbf{FES}(\beta[u_g]\,|^{V_g})$ is recursively calculated. In Case 1b (Figure 5b), some scenarios in A_h are not covered by $\mathbf{FES}(A_g)$, which means that a scenario that distinguishes H from G can be found in A_h. Therefore, $\beta[u_h]\,|^{V_h}$ is expanded to impose more restrictions to find a scenario in the shaded area. In Case 1a (Figure 5c), no scenarios in A_h are covered by $\mathbf{FES}(A_g)$, which means every scenario in A_h can be used to distinguish H from G. Therefore, the external events generated by a scenario in A_h can be used as a test case.

Let $\beta[s]$ be H, and $\beta[F]$ be G. Using the result of the equations above, $\beta[s]$ – $\mathbf{FES}(\beta[F])$ can be computed by the following algorithm.

1. Let there be a quintuple $\langle u, p, u', p', c\rangle$, where u and u' are assertions, p and p' are states, and c is 1a, 1b, or 2 with respect to Cases 1a, 1b, or 2 above. Let Q be a queue that initially contains $\langle s, \mathit{true}, u', \mathit{true}, 2\rangle$ for every state u' in F.
2. Get an element $\langle u, p, u', p', c\rangle$ from Q and check the following conditions.
   (a) If c is in Case 1a, check if there is any element, say $\langle v, q, v', q', d\rangle$ in Q such that $p = q$. If there is none, the corresponding path of assertion p generates a test case. Exit the program and return p.
   (b) If c is in Case 1b, for every outgoing transition t from state u, put $\langle v_t, (p \wedge p_t), u', p', d\rangle$ into Q, where v_t is the state t pointing to, p_t is the assertion imposed by the action of t, and d is Case 1a, 1b, or 2, based on the relation between $\{p \wedge p_t\}$ and $\mathbf{FES}(\{p'\})$.
   (c) If c is in Case 2, for every outgoing transition t from state u, and for every outgoing transition t' from state u', put $\langle v_t, (p \wedge p_t), v'_t, (p' \wedge p'_t), d\rangle$ into Q, where v_t and v'_t are the states t and t' pointing to, respectively; p_t and p'_t are the assertions imposed by the actions of t and t', respectively; and d is Case 1a, 1b, or 2, based on the relation between $\{p \wedge p_t\}$ and $\mathbf{FES}(\{p' \wedge p'_t\})$.
3. Repeat Step 2.

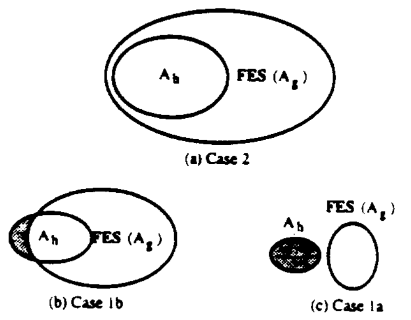


Figure 5: Illustration of the three cases of $K_{h,g}$

The idea of the algorithm can also be illustrated by Figure 5. Initially, every element in Q belongs to Case 2, which is Figure 5a. When $\beta[u]$ and $\beta[u']$ are expanded, the algorithm imposes more restrictions and less scenarios will satisfy the new restrictions. This makes the circles in Figure 5a "shrink." Eventually, the figure will become either Figure 5b or Figure 5c. If it is Figure 5c, a test scenario is found. Otherwise, the algorithm simply expand $\beta[u]$, such that the A_h circle in Figure 5b shrinks toward the shaded area and becomes Figure 5c.

Basically, it is an unsolvable problem as to whether two Turing-equivalent machines generate the same output. However, it is reasonable to assume that each transition can be finished in a bounded amount of time once it started, so that a limit can be set to the maximum number of expansions that can be performed. If the number of expansions exceeds this limit, and still no test cases can be found, we simply dictate that such faults are untestable. Noting that finding a test case for FSM has been proven by Yannakakis and Lee to be PSPACE-complete [20], we expect finding a test case for EFSM is at least as hard as the former. That is, there is unlikely any efficient algorithm besides exhaustive search. Due to the intractable nature of this problem, one can only rely on heuristic approaches to improve the performance. Some useful rules-of-thumb for the heuristic approach are discussed in Section 4.5.

4.4 Test Case Generation for Multiple Modules

Estelle allows several modules to be executed in parallel, communicating with each other through internal interaction points. Therefore, a specification of multiple modules describes a system containing several communicating EFSMs. To test such a system, the behavior of the system is defined as follows:

Definition 5 *If there are n modules, $M_1, M_2, ..., M_n$, in a specification, and let s_i be a state in module M_i, then*

$$\beta[s_1, s_2, ..., s_n] = \beta[s_1] \cap \beta[s_2] \cap \cdots \cap \beta[s_n]$$

Definition 6 *Let M be a specification that contains modules M_1, M_2, ..., M_n. Then,*

$$\beta[\![M]\!] = \bigcup_{s_1}\bigcup_{s_2}\cdots\bigcup_{s_n}\beta[s_1, s_2, ..., s_n]$$

where s_i is a state in M_i.

The behavior of a multiple-module system can be derived in the same way as the behavior of a single EFSM, except that every module has its own auxiliary variables τ and κ. It is assumed that different modules do not share the same variable names. Such a name conflict can easily be resolved by appending module names to the variable names. Figure 6 is a simple send-and-receive protocol, and its behavior is shown as follows:

Two scenarios for the sender and the receiver of the protocol can be derived as

$$\begin{aligned}\beta[S] = \;& \{(\nu c_s = send(\nu d)) \wedge (\nu d = \nu\kappa_s) \wedge (\nu\tau_s = \nu c_s) \wedge \\ & (\nu c_i = msg(\nu d)) \wedge (\nu^2\tau_s = \nu c_i)\} \cap \\ & \beta[S] |^{\{\nu c_s, \nu d, \nu\kappa_s, \nu^2\tau_s, \nu c_i\}} \\ \beta[R] = \;& \{(\nu c_i = msg(\nu b)) \wedge (\nu b = \nu\kappa_r) \wedge (\nu\tau_r = \nu c_i) \wedge \\ & (\nu c_r = rec(\nu b)) \wedge (\nu^2\tau_r = \nu c_r)\} \cap \\ & \beta[R] |^{\{\nu c_r, \nu b, \nu\kappa_r, \nu^2\tau_r, \nu c_i\}}\end{aligned}$$

Therefore,

$$\begin{aligned}& \beta[S, R] = \beta[S] \cap \beta[R] = \\ & \{(\nu\tau_s = \nu c_s = send(\nu d)) \wedge (\nu d = \nu\kappa_s)\} \cap \\ & \{(\nu^2\tau_s = msg(\nu d) = \nu c_i = msg(\nu b) = \nu\tau_r) \wedge (\nu b = \nu\kappa_r)\} \cap \\ & \{(\nu^2\tau_r = \nu c_r = rec(\nu b))\} \cap \\ & \beta[S] |^{\{\nu c_s, \nu d, \nu\kappa_s, \nu^2\tau_s, \nu c_i\}} \cap \beta[R] |^{\{\nu c_r, \nu b, \nu\kappa_r, \nu^2\tau_r, \nu c_i\}}\end{aligned}$$

Note that channel c_i relates the output of the sender to the input of the receiver. Since $\nu c_i = msg(\nu d)$ in the sender and $\nu c_i = msg(\nu b)$ in the receiver, νd equals to νb. Note also that each module has its own variables τ and κ. The external events are those τ's which record events on the channels that connect external IPs. For example, $\nu\tau_s$ and $\nu^2\tau_r$ are external events since $\nu\tau_s = \nu c_s$, $\nu^2\tau_r = \nu c_r$, and c_s and c_r are connected to external IPs. The order of external events is preserved by variables τ's. For instance, because $\nu\tau_s$ and $\nu\tau_r$ precede $\nu^2\tau_s$ and $\nu^2\tau_r$, respectively, and because $\nu^2\tau_s = \nu\tau_r$, $\nu\tau_s$ precedes $\nu^2\tau_r$.

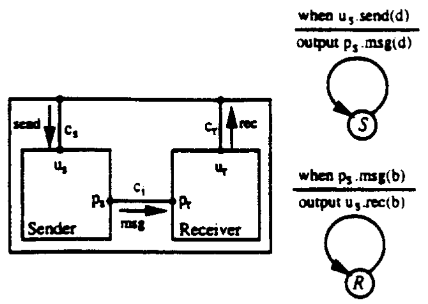


Figure 6: A simple send-and-receive protocol

The mutants of a multi-module system can be specified by Estelle, and their behavior can also be described by the behavior model. Hence, the test case for a multi-module specification can be generated using the same algorithm for a single module specification.

Estelle allows for dynamic creating and destroying of a module instance through statements **init** and **release**, respectively. It can also dynamically create and destroy a channel between two IPs by issuing **connect**, **disconnect**, **attach**, and **detach** statements. Therefore, an unique name must be assigned to each newly created object. For example, if statement "**init** *mv* **to** *body*" is issued (where *mv* is a "module variable" and *body* is a "module body"), a variable x in module body "*body*" should be referred to as $mv.n.x$, where n is the number of times mv is initialized to a module body. If statement "**connect** p **to** q" is issued (where p and q are IPs), a new variable for the channel between p and q is created with a name such as $C_{p,q}$. With the assurance that each object has its own name, the algorithm in Section 4.3 can run without a hitch.

4.5 Heuristic Search Methods

Since the conformance of two Turing-equivalent machines is generally an unsolvable problem, it is unlikely to find a test case generation algorithm without using an exhaustive search. However, heuristics can be used to improve the performance of the test case searching process.

From our experience, there are some guidelines that can be used to improve the performance of the algorithm in Section 4 dramatically:

1. Expand those items in queue Q that lead to faulty transitions first.
2. After executing a faulty transition, traverse through those paths which output the values of the variables that appear in the faulty transition first.
3. If there are more than one path in the abov-. traverse the shortest one first.

Using these guidelines, the algorithm described in Section 4.3 can be modified as follows:

1. Let Q be a priority queue.
2. For every element $\langle u, p, u', p', c\rangle$ that is to be added to Q, check the following:

(a) If the corresponding path, say P', of predicate p' does not contain any faulty transition, set the priority value of the element to be the length of the shortest path from the last transition of P' to a faulty transition in F.

(b) If the corresponding path P' of predicate p' contains a faulty transition, set the priority value of the element to be the length of the shortest path between the last transition in P' and a transition that refers to any variable used in the the faulty transition.

3. To select an element in Q, choose the one with the lowest priority value.

5 Conclusion

In this paper, an automatic test case generation method for an ISO standard specification language, Estelle, is presented. The method compares the behavior of a specification to the behavior of a given fault model. Based on their difference, test cases are mechanically derived. Unlike other conformance testing methods that apply only to fixed fault models, the proposed method is able to generate a test case that detects possible implementation errors specified by a given fault model. Therefore, one can test those critical faults and obtains confidence in the coverage of such faults. When time is critical and resources are limited, it becomes very important to be able to test the most critical faults and frequently executed transitions first.

While most of the existing methods concern only generating test cases for a single entity, the proposed method is able to deal with those specifications that contain multiple modules. Treating channels between two modules as variables, the method transforms the I/O statements into assignment statements and derives test cases by using the same algorithm used to generate test cases for a single module.

Test case generation has been proven to be at least PSPACE-hard. To improve the performance of the proposed algorithm, heuristics are introduced. The guidelines suggested in this paper significantly reduce the number of states exploited, thereby improving the performance of the test case generation processes.

In summary, the method proposed in [16] is extended to generating test cases for Estelle. It is also extended to dealing with multiple modules. In addition, some heuristics are suggested to improve the performance of the test case generation process. Currently, a test case generator for EFSM with given fault models has been developed, and an extension of it to Estelle is under consideration.

References

[1] T. Chow, "Testing software design modeled by finite-state machines," *IEEE Trans. on Software Engineering*, vol. SE-4, no. 3, pp. 178–187, March 1978.

[2] G. Gönenç, "A model for the design of fault detection experiments," *IEEE Trans. on Computers*, vol. C-19, no. 6, pp. 551–558, June 1970.

[3] S. Naito, "Fault detection for sequential machines by transition tours," in *Proc. 11th IEEE Symp. on Fault Tolerant Computing*, pp. 238–243, 1981.

[4] K. Sabnani and A. Dahbura, "A protocol test generation procedure," *Computer Networks and ISDN Systems*, vol. 15, pp. 285–297, 1988.

[5] G. v. Bochmann and J. Gecsei, "A unified method for the specification and verification of protocols," in *Proc. IFIP Congress '77*, pp. 229–234, 1977.

[6] S. Budkowski and P. Dembinski, "An introduction to Estelle: A specification language for distributed systems," *Computer Networks and ISDN Systems*, vol. 14, no. 1, pp. 3–23, 1987.

[7] T. Bolognesi and E. Brinksma, "Introduction to the ISO specification language LOTOS," *Computer Networks and ISDN Systems*, vol. 14, pp. 25–59, 1987.

[8] E. Brinksma, "A theory for the derivation of tests," in *The Formal Description Technique LOTOS* (P. van Eijk, C.A.Vissers, and M. Diaz, eds.), pp. 235–247, Elsevier Science Publishers B.V. (North-Holland), 1989.

[9] R. Langerak, "A testing theory for LOTOS using deadlock detection," in *Proc. 10th IFIP Symp. on Protocol Specification, Testing, and Verification*, pp. 87–98, 1990.

[10] B. Sarikaya, G. V. Bochmann, and E. Cerny, "A test design methodology for protocol testing," *IEEE Trans. on Software Engineering*, vol. SE-13, no. 5, pp. 518–531, May 1987.

[11] H. Ural, "Test sequence selection based on static data flow analysis," *Computer Communications*, vol. 10, no. 5, pp. 234–242, 1987.

[12] H. Ural and B. Yang, "A test sequence selection method for protocol testing," *IEEE Trans. on Communications*, vol. 39, no. 4, pp. 514–523, April 1991.

[13] C.-J. Wang and M. T. Liu, "Axiomatic test sequence generation for extended finite state machines," in *Proc. 12th International Conference on Distributed Computing Systems*, pp. 252–259, June 1992.

[14] C.-J. Wang and M. T. Liu, "A test suite generation method for extended finite state machines using axiomatic semantics approach," in *IFIP Trans. Protocol Specification, Testing, and Verification, XII*, pp. 29–43, North-Holland, 1992.

[15] G. v. Bochmann, A. Das, R. Dssouli, M. Dubuc, A. Ghedamsi, and G. Luo, "Fault models in testing," in *Protocol Test Systems, IV*, pp. 17–30, Elsevier Science Publisher B.V. (North-Holland), 1992.

[16] C.-J. Wang and M. T. Liu, "Generating test cases for EFSM with given fault models," in *Proc. IEEE INFOCOM '93*, pp. 774–781, March 1993.

[17] L. A. Clarke and D. J. Richardson, "Applications of symbolic evaluation," *Journal of Systems and Software*, vol. 5, pp. 15–35, 1985.

[18] J. C. King, "Symbolic execution and program testing," *CACM*, vol. 19, no. 7, pp. 385–394, July 1979.

[19] F. G. Pagan, *Formal Specification of Programming Languages: A Panoramic Primer*, ch. 4, pp. 193–215, Prentice-Hall, Inc., 1981.

[20] G. J. Holzmann, *Design and Validation of Computer Protocols*, ch. 9, pp. 198–199, Prentice-Hall, Inc., 1991.

B. H. Yoon, M. T. Liu, K. Y. Lee and Y. M. Kim,
"The Knockout Switch Under Nonuniform Traffic,"
IEEE Trans. on Communications,
Vol. 43, No. 4, pp. 2149–2156, June 1995.

The Knockout Switch Under Nonuniform Traffic

Hyunsoo Yoon, Ming T. (Mike) Liu, *Fellow, IEEE*, Kyungsook Y. Lee, and Young Man Kim

Abstract—**The Knockout Switch [14] is a nonblocking, high-performance switch suitable for broadband packet switching. It allows packet losses, but the probability of a packet loss can be kept extremely small in a cost-effective way. The performance of the Knockout Switch was analyzed under uniform traffic [14].**

In this paper, we present a new, more general analytic model of the Knockout Switch, which enables us to evaluate the Knockout Switch under nonuniform traffic. The new model also incorporates the effects of a concentrator and a shared buffer on the packet loss probability. Numeric results for nonuniform traffic patterns of interest are presented.

I. Introduction

Broadband packet switching technologies are maturing rapidly for Broadband Integrated Services Digital Networks. One key element necessary for providing broadband services is a high-speed, high-capacity packet switch interconnecting a large number of nodes. Many communication architectures proposed for broadband packet switching are based on self-routing networks such as banyan networks [5], [9], [12].

The Knockout Switch was proposed for high-speed local and metropolitan area networks, a multiprocessor interconnection, and local or toll switches for integrated traffic loads [2], [7], [12]. The Knockout Switch employs a complete connection to eliminate the internal blocking problem, which has been the major obstacle to the high performance in the traditional $O(N\log N)$ multistage switches such as banyan networks [6], [10], [13], where N is the switch size.

As shown in Fig. 1, the $(N \times N)$ Knockout Switch uses N input broadcast buses and N bus interfaces to achieve a complete interconnection of the inputs and outputs. The packets arriving at each input enters the input broadcast bus. Associated with each output is a bus interface module listening to all N input broadcast buses.

A bus interface consists of N packet filters, a concentrator, and a shared buffer, as shown in Fig. 2. Each packet filter checks the destination of the incoming packet and allows the packet to enter the concentrator only if it is addressed to the output the packet filter is connected to. The concentrator selects up to $L(L \ll N)$ packets by the knockout contention scheme [14] and feeds them into the shared buffer which acts as an FIFO queue with L inputs and 1 output. Since each concentrator takes $O(N)$ switches [14] and there are N such concentrators, the hardware complexity of the Knockout Switch is $O(N^2)$.

If more than L packets destined for the same output arrive simultaneously at the N inputs, the packets losing the contention are dropped in the concentrator. It was shown that the packet loss probability at the $N{:}L$ concentrator can be kept below 10^{-6} with $L = 8$ regardless of the switch size N under uniform traffic [14]. With any practical networks, the packet losses of this magnitude are inevitable due to transmission line errors, buffer overflows, and network failures. In any case, the lost packets can be recovered by the end-to-end protocol. Furthermore, the lost packet probability can be made as low as desired by adding more concentrator outputs.

With uniform traffic, the N inputs have the same input load, and the input load on an input is distributed equally over all the outputs. It would be interesting to investigate the Knockout Switch under nonuniform traffic to see whether the packet loss probability can still be kept extremely small with a fixed number of concentrator outputs for an arbitrary large N. This constitutes the main objective of this paper.

The analysis of the Knockout Switch in [14] consists of two separate parts: one for the immediate lost packet probability at the concentrator due to simultaneous packet arrivals, and the other for the probability of the shared buffer overflow over time. Both require the uniform traffic assumption.

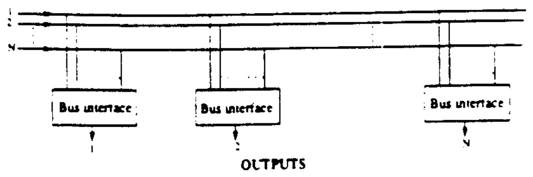


Fig. 1. The Knockout Switch interconnection fabric [13].

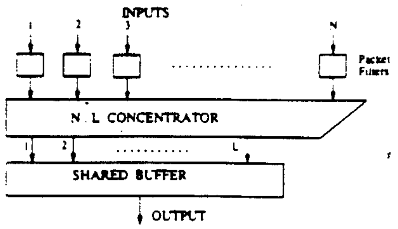


Fig. 2. Bus interface [13].

Paper approved by G. P. O'Reilly, the Editor for Communications Switching of the IEEE Communications Society. Manuscript received September 10, 1992; revised October 4, 1993. This work was supported by the U.S. Army CECOM, Ft. Monmouth, NJ, under Contracts DAAL03-91-G-0093 and DAAL03-92-G-0184. The views, opinions, and/or findings contained in this paper are those of the authors and should not be construed as an official Department of the Army position, policy, or decision. The work of K. Y. Lee was supported in part by the National Science Foundation under Grant MIP-9210605.

H. Yoon is with the Department of Computer Science, KAIST, Taejon 305-701, Korea.

M. T. Liu is with the Department of Computer and Information Science, The Ohio State University, Columbus, OH 43210 USA.

K. Y. Lee is with the Department of Mathematics and Computer Science, University of Denver, Denver, CO 80208 USA.

Y. M. Kim is with Central Laboratory, Mitsubishi Materials Corporation, Omiya City, Japan.

IEEE Log Number 9410436.

In this paper, we present a more general analytic model of the Knockout Switch which integrates the effects of the concentrator and the shared buffer on the lost packet probability, and allows any traffic patterns.

In Section II, we briefly discuss nonuniform traffic. In Section III, we present a new analysis of the Knockout Switch under any traffic patterns. The numeric results obtained from the analysis for several nonuniform patterns of interest are presented in Section IV. Conclusions are given in Section V.

II. Nonuniform Traffic

For the input load distribution on the $(N \times N)$ switch, define an $(N \times N)$ input load matrix $I = \{\rho_{ij}\}$, where element $\rho_{ij}(1 \leq i,j \leq N)$ represents the input load from input i to output j. In other words, ρ_{ij} represents the probability of a packet arriving at input i destined for output j. Thus, the sum of row i of the input load matrix I represents the total load on input i and the sum of column j represents the total load to output j. Uniform traffic denotes the case when all the elements of the load matrix have the same value, and nonuniform traffic refers to any traffic patterns which are not uniform. Among all the conceivable nonuniform traffic patterns, we are interested in the patterns which have been investigated in the literature such as "hot-spot" [11] and "point-to-point" traffic [13].

The hot-spot traffic is a nonuniform traffic pattern consisting of a single output of higher access rate (hot-spot) superimposed on a background uniform traffic. For multistage interconnection networks with the internal path sharing among different inputs/outputs, even a moderate hot-spot access rate was found to significantly degrade all output access, not just access to the hot-spot.

The effects of hot-spot traffic on the Knockout Switch is expected to be much less severe, since the Knockout Switch is based on a complete connection with no path interference among different input/output nodes. In the Knockout Switch, the output port loading is the determining factor for its performance. The hot-spot traffic does result in a higher output loading at the hot-spot itself, and its effects on the output buffer size and the number of concentrator outputs need to be examined.

Since the Knockout Switch employs a complete connection and provides buffers at the outputs for the packets arriving at the same output, it provides the lowest latency in any switching arrangement [8]. However, since packets can be lost within the switch, the Knockout Switch should have extremely low packet loss probabilities, and the Knockout Switch does have such a property under uniform traffic. Therefore, the main objective of the analysis is to obtain lost packet probabilities of the Knockout Switch under various nonuniform traffic patterns to evaluate its effectiveness under those traffic patterns.

III. Analysis of the Knockout Switch

Our analytic model of the $(N \times N)$ Knockout Switch is an approximate Markov chain widely used in the literature [6], [15]. The model assumes the following.

- The switch operates synchronously with a fixed stage cycle.
- Packets arrive independently at each network input and the time interval between successive packet arrivals is geometrically distributed.
- Packets are of a fixed length.
- The network is error-free.

A. Variables

N Knockout Switch size.
m buffer size.
L threshold (number of the concentrator outputs), $L \leq m$.
$I = \{\rho_{ij}\}$ $(N \times N)$ input load matrix; ρ_{ij} represents the input load from input i to output j. $(1 \leq i,j \leq N)$.
q_j^k probability that k packets arrive at the concentrator of output j during each stage cycle, $(0 \leq j,k \leq N)$.
r_j^k probability that k packets succeed to pass through the concentrator of output j during each stage cycle, $(0 \leq k \leq L, 1 \leq j \leq N)$.
$p_j^k(t)$ probability that there are k packets in the buffer of output j at the beginning of stage cycle t, $(0 \leq k \leq m, 1 \leq j \leq N)$.
p_j^k steady state value of $p_j^k(t)$.
T_j throughput of output j, $(1 \leq j \leq N)$.
T^i throughput of input i, $(1 \leq i \leq N)$.
D_j average delay of a packet through output j, $(1 \leq j \leq N)$.
D average delay of a packet through the switch.

B. Equations

Given the input load matrix $I = \{\rho_{ij}\}$ for the Knockout Switch, the probability that k packets arrive at the concentrator of output j during a stage cycle, q_j^k, under the assumption that packets arrive at each input independently, is given by

$$q_j^k = \sum_{|S|=k} \prod_{i \in S} \rho_{ij} \prod_{i \in \bar{S}} (1 - \rho_{ij}), \qquad 1 \leq j,k \leq N. \tag{1}$$

That is, it is a summation of the probabilities of all the cases when some k inputs have packets destined for output j and the other $N - k$ inputs do not.

The input rate and the output rate are the same for the concentrator of output j as long as the capacity of the concentrator is not fully utilized. When the capacity is fully utilized with L or more packets arriving each cycle, the concentrator outputs L packets in each cycle.

$$r_j^k = q_j^k, \qquad 0 \leq k \leq L-1,$$
$$r_j^L = \sum_{k=L}^{N} q_j^k. \tag{2}$$

Next we consider the state transition of a buffer of size m for output j. The output consumes one packet in each cycle as long as the buffer is not empty. A packet entering an empty buffer passes through the buffer to the output. In steady state,

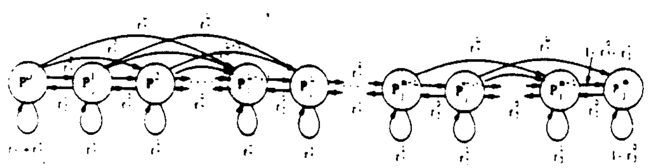


Fig. 3. State transition diagram of the buffer of size m for output j.

from the transition diagram in Fig. 3.

$$p_j^0 \cdot (r_j^0 + r_j^1) + p_j^1 \cdot r_j^0 = p_j^0$$
$$\sum_{a=0}^{i+1} p_j^a \cdot r_j^{i-a+1} = p_j^i, \qquad 1 \le i \le L-1 \qquad (3)$$
$$\sum_{a=i-L+1}^{i+1} p_j^a \cdot r_j^{i-a+1} = p_j^i, \qquad L \le i \le m-1$$
$$\sum_{i=0}^{m} p_j^i = 1.$$

As the performance measures, throughput at output j, T_j, throughput at input i, T^i, average packet delay through output j, D_j, and average packet delay through the switch, D, are used.

The only possible situation that a buffer can not provide a packet for its output is when there is no packet in the buffer at the beginning of a cycle and no packets are coming into the buffer during the cycle. The throughput at output j, T_j, can also be expressed as the packet input rate to buffer j minus the probability of a packet discarded in the buffer due to buffer overflow. Since the maximum number of packets entering the buffer per cycle is L and one packet is consumed by the output per cycle, the number of packets in the buffer can grow at most by $(L-1)$ per cycle. Thus the buffer overflow might happen when the buffer already has $m-(L-2)$ or more packets at the beginning of a stage cycle with $(L-2)$ or less free slots. For these buffer conditions, the buffer with k packets in it does overflow if k' packets enter the buffer, $m-(k-2) \le k' \le L$, losing $k' - \{m-(k-1)\}$ packets. Thus,

$$\begin{aligned} T_j &= 1 - p_j^0 \cdot r_j^0 \\ &= \sum_{k=1}^{L} k \cdot r_j^k - \sum_{k=m-(L-2)}^{m} p_j^k \\ &\quad \cdot \left[\sum_{k'=m-(k-2)}^{L} \{(k' - (m-k+1)\} \cdot r_j^{k'} \right]. \end{aligned} \qquad (4)$$

The throughput at input i, T^i, is the sum of the fractions of throughputs at all outputs originated from input i.

$$T^i = \sum_{j=1}^{N} \rho_{ij} \cdot \frac{T_j}{\sum_{i'=1}^{N} \rho_{i'j}}. \qquad (5)$$

When a packet enters the bth slot of the buffer for output j, the delay of the packet will be b cycles. When k' packets pass through the concentrator at output j whose buffer has k packets at the beginning of the cycle, they are stored in the slots between k and $k+k'-1$. So the delay at output j, D_j, can be expressed as following.

$$D_j = \sum_{k=0}^{m} \left(p_j^k \cdot \sum_{k'=1}^{L} \left(r_j^{k'} \cdot \sum_{b=1}^{k'} \frac{\min\{m, k+b-1\}}{k'} \right) \right). \qquad (6)$$

The input load to output j, ρ_j, is simply the sum of all input loads destined for output j.

$$\rho_j = \sum_{i=1}^{N} \rho_{ij}. \qquad (7)$$

The *input-uniform* traffic pattern is an important subcase of nonuniform traffic in which the input load profile of all input nodes are the same. Formally, for input-uniform traffic,

$$\rho_{ij} = \rho_{i'j} = \frac{\rho_j}{N}, \qquad 1 \le i, i', j \le N. \qquad (8)$$

For input-uniform traffic, the equations for q_j^k and T^i can be reduced to the following.

$$q_j^k = \binom{N}{k} \left(\frac{\rho_j}{N}\right)^k \left(1 - \frac{\rho_j}{N}\right)^{N-k} \qquad (9)$$
$$T^i = \frac{\sum_{j=1}^{N} T_j}{N}. \qquad (10)$$

IV. Analytic Results

In this section, first we use uniform traffic to verify the accuracy of the new analytic model by comparing the numeric results obtained from the new model to the known results for uniform traffic in [14].

Then, numeric results for some nonuniform traffic including hot-spot traffic [11] and point-to-point traffic [14] are presented.

A. Uniform Traffic

The lost packet probability versus input load are plotted in Fig. 4 with varying threshold L for the (1024×1024) Knockout Switch. The buffer length m is set to 40. It shows that with $L = 8$, the lost packet probability less than 10^{-6} can be achieved for the input loads upto 0.84.

In Fig. 5, the effect of the buffer size, m, on the lost packet probability is shown for the (1024×1024) Knockout Switch with $L = 8$. It shows that with $m = 40$, the lost packet probability less than 10^{-6} can be obtained for the input loads up to 0.84.

In Fig. 6, lost packet probabilities are plotted for different network sizes with $m = 40$ and $L = 8$. It shows that lost packet probabilities stay under 10^{-6} for input loads up to 0.84 as the network size is increased.

All the results from Figs. 4-6 confirm the earlier results [14] verifying the accuracy of our new analytic model.

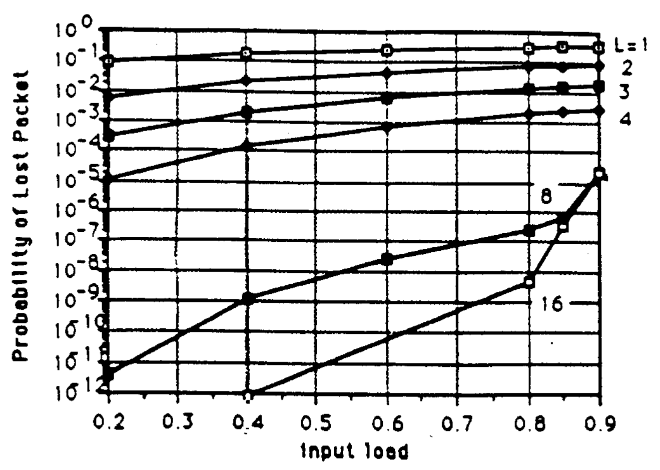


Fig. 4. Lost packet probability versus input load ($N = 1024, m = 40$); uniform traffic.

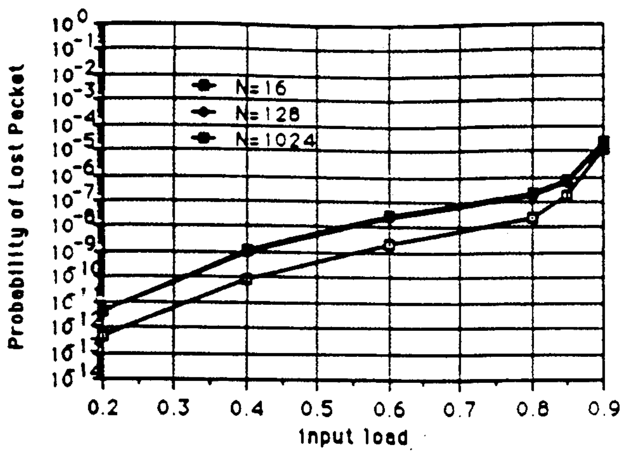


Fig. 6. Lost packet probability versus input load ($m = 40, L = 8$); uniform traffic.

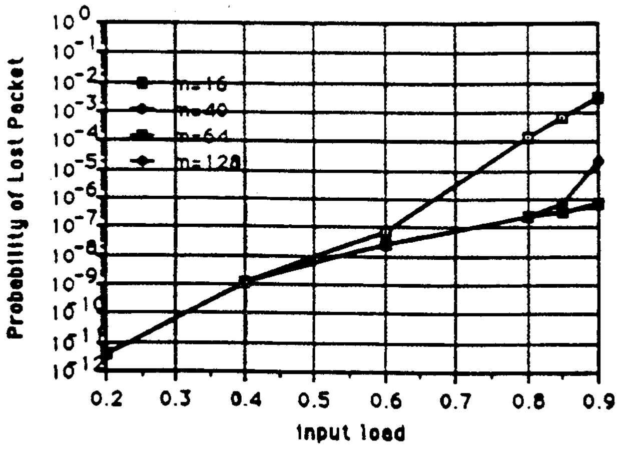


Fig. 5. Lost packet probability versus input load ($N = 1024, L = 8$); uniform traffic.

B. Hot-Spot Traffic

Hot-spot traffic refers to a traffic pattern where many sources try to communicate with one destination (hot-spot) virtually at the same time. The hot-spot traffic pattern could occur in many application areas. For example, many callers may compete to reach a particular subscriber in a telephone network. Hot-spot traffic occurs most notably in shared memory multiprocessor systems, in which processors and memory modules are interconnected through an interconnection network. In such a system, any variable shared by many processors creates a memory contention at the associated memory module. Those shared variables, called hot-spots, could be locks for the process synchronization. It is shown in [11] that even a small percentage of hot-spot requests can significantly degrade the performance of a system based on a banyan network.

Hot-spot traffic can also occur in computer networks. For example, many nodes may report synchronously to one node, e.g., the network control center, for administrative purposes. Another example is a local area network consisting of many diskless computer systems and a single file server. In such a local area network [1], all the computers on the network run a process at midnight to resynchronize their clocks. A diskless system gets all of its files over the network from a central file server. Thus, at midnight, all the diskless computers have to load all the programs and data files needed for its execution from the file server, resulting in serious hot-spot contention at the file server. Considering that one objective of the computer networks is resource sharing, and any kind of sharing has a potential of contention, any computer network has a potential of hot-spot traffic.

Since the applications of the Knockout Switch include the LAN's and multiprocessor interconnections where hot-spot traffic may occur, it is natural to investigate the performance of the Knockout Switch under hot-spot traffic. We use the hot-spot traffic model in [11]: a nonuniform traffic pattern consisting of a single hot-spot of a higher access rate superimposed on the background uniform traffic. Thus the hot-spot traffic is an input-uniform traffic pattern. Let h be a fraction of packets directed to the hot-spot. Then, ρ, the packet arrival rate at an input can be expressed as $\rho = h\rho + (1-h)\rho$, where $h\rho$ packets are directed to the hot-spot and $(1-h)\rho$ packets are directed uniformly to the N outputs. With this hot-spot traffic model, the probability that k packets arrive at the hot-spot's concentrator, q_h^k, is

$$q_h^k = \binom{N}{k}\left[h\rho + \frac{(1-h)\rho}{N}\right]^k \left[1 - h\rho - \frac{(1-h)\rho}{N}\right]^{N-k}. \tag{11}$$

The numeric results for hot-spot traffic are given in Figs. 7–13. For these measurements, the maximum hot-spot load, $h_{\max}$, was chosen so as not to overload the output which is the hot-spot:

$$\{(1-h)\rho + Nh\rho\} \simeq \{\rho + Nh\rho\} \leq 1.0, \tag{12}$$

$$h_{\max} \simeq \frac{1-\rho}{N\rho}. \tag{13}$$

In Table I, $h_{\max}$ for different input loads and network sizes are listed.

Figs. 7–10 plot lost packet probability vs. hot-spot load with various input loads for network sizes 1024, 256, 128 and 64, respectively. The buffer size m was set to 40, and the threshold L was set to 8.

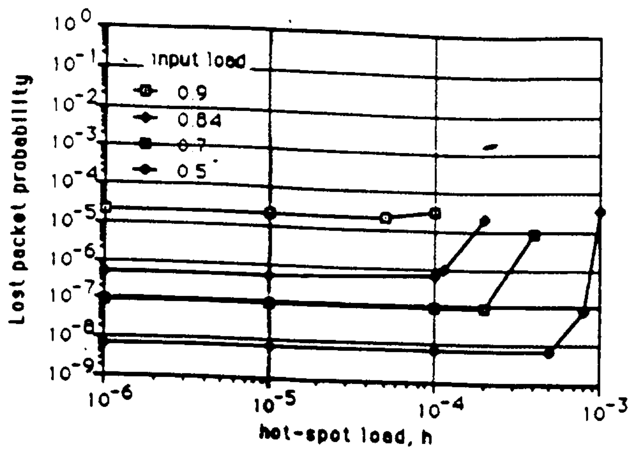


Fig. 7. Lost packet probability versus hot-spot load ($N = 1024$, $m = 40$, $L = 8$).

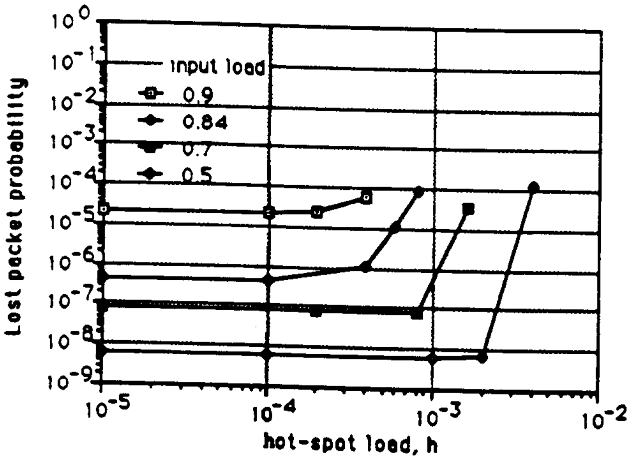


Fig. 8. Lost packet probability versus hot-spot load ($N = 256$, $m = 40$, $L = 8$).

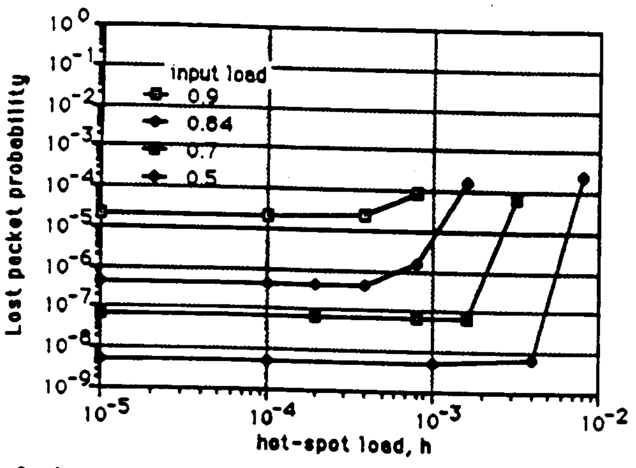


Fig. 9. Lost packet probability versus hot-spot load ($N = 128$, $m = 40$, $L = 8$).

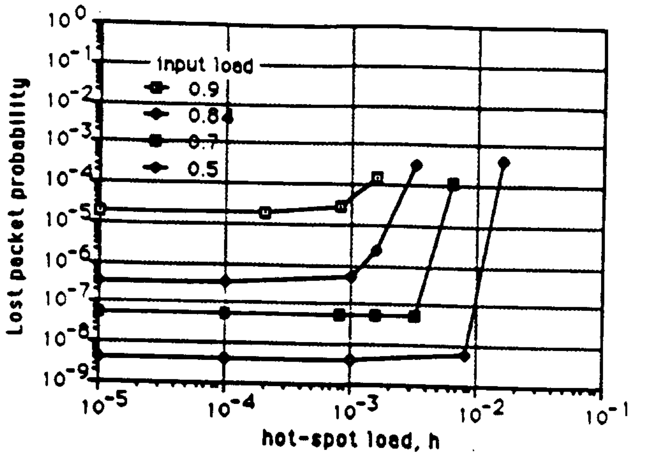


Fig. 10. Lost packet probability versus hot-spot load ($N = 64$, $m = 40$, $L = 8$).

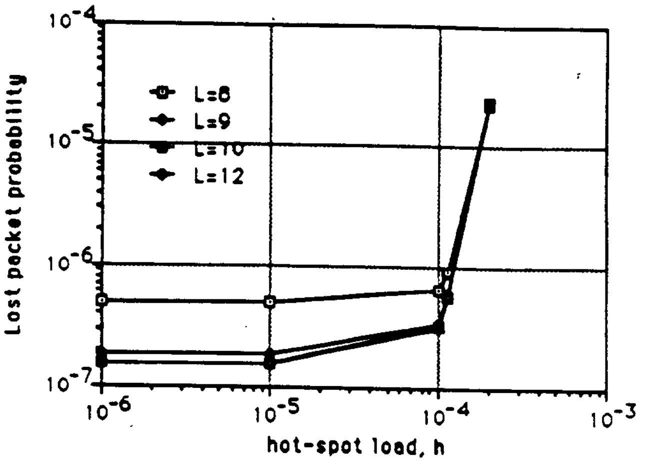


Fig. 11. Lost packet probability versus hot-spot load ($N = 1024$, input load = 0.84, $M = 40$).

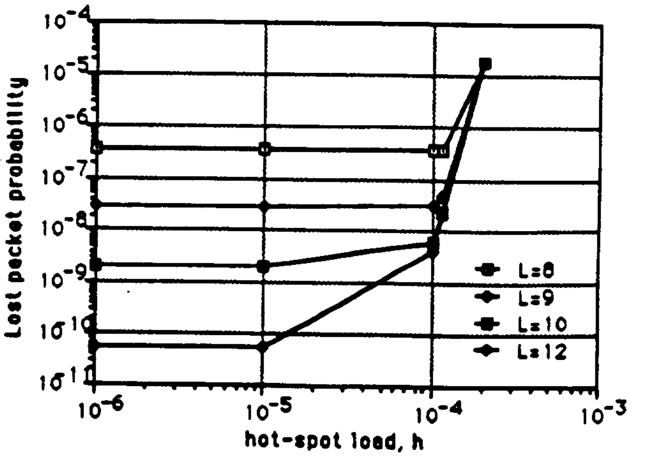


Fig. 12. Lost packet probability versus hot-spot load ($N = 1024$, input load = 0.84, $M = 64$).

As expected, with $\rho > 0.84$ the lost packet probabilities exceed 10^{-6} in all cases. The lost packet probabilities also exceed 10^{-6} with the hot-spot load approaching $h_{\max}$ even for $\rho \leq 0.84$. Given the network size, N, and the input load, ρ, there is a value for the hot-spot load, h, which keeps the lost packet probability under 10^{-6} with $\rho \leq 0.84$. These values are tabulated in Table II.

The effects of the increased buffer size and threshold on the lost packet probability are examined in Figs. 11–13. The network size was set to 1024, and the input load was set to 0.84.

For Fig. 11, the buffer size was set to 40, and the threshold was varied from 8 to 12. Although the lost packet probabilities

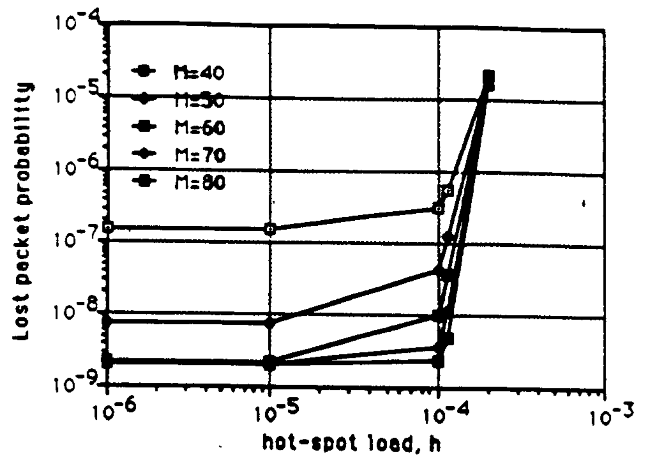


Fig. 13. Lost packet probability versus hot-spot load ($N = 1024$, input load $= 0.84$, $L = 10$).

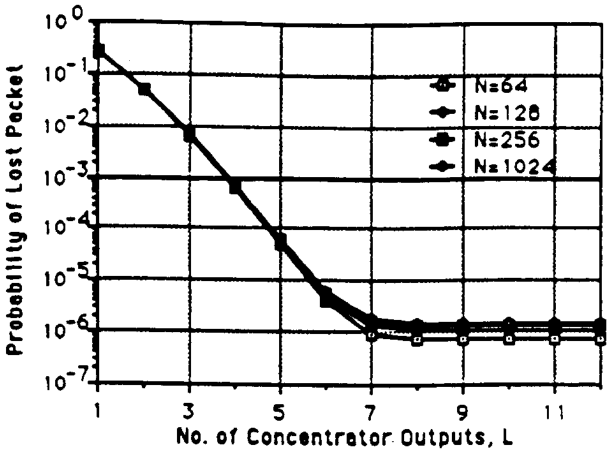


Fig. 14. Probability of lost packet versus threshold L; point-to-point traffic (input load $= 0.45$, $m = 40$).

TABLE I
MAXIMUM h NOT TO OVERLOAD THE HOT-SPOT

input load ρ \ N	1024	256	128	64
0.9	0.01	0.04	0.08	0.16
0.84	0.02	0.08	0.16	0.32
0.7	0.04	0.16	0.32	0.64
0.5	0.1	0.4	0.8	1.6

TABLE II
VALUES OF h TO KEEP THE LOST PACKET PROBABILITY $\leq 10^{-6}$

input load ρ \ N	1024	256	128	64
0.84	0.01	0.04	0.06	0.1
0.7	0.03	0.1	0.2	0.4
0.5	0.08	0.3	0.6	1.0

decrease for low h's as the threshold increases, the value of L which keeps the lost packet probability under 10^{-6} does not change for larger threshold values.

The same trend is true in Fig. 12 which is similar to Fig. 11 except that the buffer size was increased to 64.

For Fig. 13, the threshold was set to 10, and the buffer size was varied from 40 to 80. Again, althrough larger buffer sizes result in lower lost packet probabilities for low h's, the cutoff point of h to keep the lost packet probability under 10^{-6} does not change.

From Figs. 11–13, it can be seen that the (1024×1024) Knockout Switch is stable under hot-spot traffic with the buffer size of 40 and the threshold of 8 (the same configuration as for uniform traffic) if the hot-spot load is limited to values listed in Table II. Once the hot-spot load becomes higher than these cut-off points, the lost packet probability becomes higher than 10^{-6}. At this point, increased buffering and threshold do not help. We observed the same phenomenon for smaller network sizes, too. This suggests the importance of regulating the hot-spot load within the range not to overload the hot-spot to keep the Knockout Switch stable.

C. Point-to-Point Traffic

The second nonuniform traffic pattern of interest is mixed point-to-point/uniform traffic that has a particular significance to mixed voice, data, and video applications [13]. A point-to-point traffic exists when at least one input link of a packet switch sends all of its load to a unique output link. Such "dedicated" traffic might exist if, for example, broadband traffic (e.g., file transfer or video) is supported through the packet switch. In mixed point-to-point/uniform traffic, there is one point-to-point traffic and the rest of input/output links form uniform traffic. We may envision the traffic pattern as representing one single video channel in the presence of packetized voice calls. A banyan network has been analyzed under this traffic model and shown to perform poorly [13].

Let j be the output where point-to-point traffic arrives from an input, and ρ the input load of each input. Then the probability of k packets arriving at the concentrator of output j is given by

$$q_j^k = (1-\rho)\binom{N-1}{k}\left(\frac{\rho}{N}\right)^k\left(1-\frac{\rho}{N}\right)^{N-1-k} + \rho\binom{N-1}{k-1}\left(\frac{\rho}{N}\right)^{k-1}\left(1-\frac{\rho}{N}\right)^{N-k}. \quad (14)$$

Figs. 14–15 show, for $\rho = 0.45$, lost packet probability versus L for various N's. The buffer size was set to 40 and 80, respectively. It can be seen that to achieve the packet loss probability of less than 10^{-6}, the buffer size needs to be 80 and $L \geq 7$. Therefore, the Knockout Switch is stable under mixed point-to-point/uniform traffic, given sufficient buffers. On the contrary, it has been shown in [13] that the multistage packet switches, banyan networks, significantly degrade their performance under point-to-point traffic.

D. Some Other Nonuniform Traffic Patterns

In this subsection, we consider two other nonuniform traffic patterns which have been used in analyzing banyan networks in [3]. We call the two traffic patterns traffic patterns 1 and 2, respectively.

Traffic Pattern 1: Consider the following nonuniform traffic pattern. The load matrix I is partitioned as

$$I = \begin{bmatrix} I_1(\rho_1) & I_2(\rho_2) \\ I_2(\rho_2) & I_1(\rho_1) \end{bmatrix}.$$

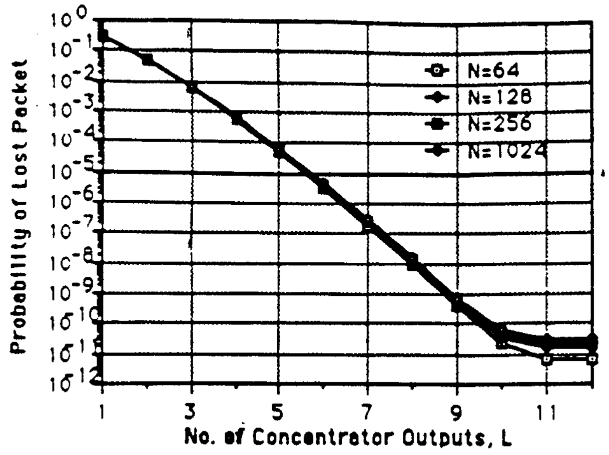


Fig. 15. Probability of lost packet versus threshold L: point-to-point traffic (input load = 0.45, m = 80).

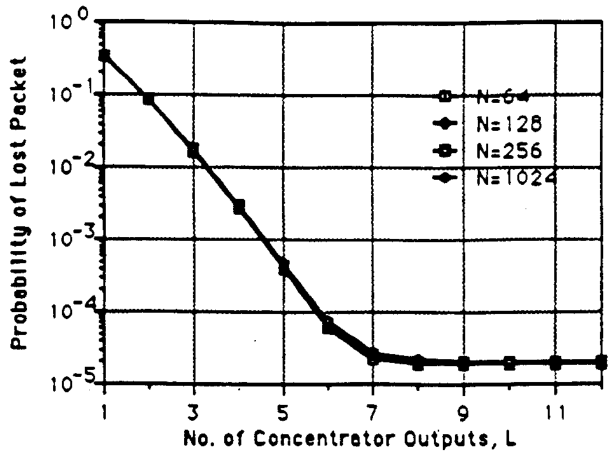


Fig. 16. Probability of lost packet versus threshold L: traffic pattern 2 (input load = 0.45, r = 1, m = 40).

Here matrix I is not uniform, but submatrices I_1 and I_2 are uniform. The row sum of I_1 is ρ_1, and the row sum of I_2 is ρ_2 such that $\rho_1 + \rho_2 = \rho$. The partition is done symmetrically: there are two equal-sized groups of sources as well as destinations. The groups are called source group 1, source group 2, destination group 1, and destination group 2, respectively. The probability of a packet originating at source group $i (i = 1, 2)$ destined for destination group $j, i \neq j$, is ρ_2 (e.g., light traffic). The probability of a packet originating from source group $i (i = 1, 2)$ destined for destination group i is ρ_1 (e.g., heavy traffic). In practice, such a traffic pattern may result when some sources communicate with some destinations (favorite groups) more frequently than with other destinations.

Under this traffic pattern, the probability of k packets arriving at the concentrator of output j of the Knockout Switch is given by

$$q_j^k = \sum_{k_1+k_2=k} \binom{N/2}{k_1}\left(\frac{2\rho_1}{N}\right)^{k_1}\left(1-\frac{2\rho_1}{N}\right)^{N/2-k_1} \cdot \binom{N/2}{k_2}\left(\frac{2\rho_2}{N}\right)^{k_2}\left(1-\frac{2\rho_2}{N}\right)^{N/2-k_2}. \tag{15}$$

Although the traffic pattern in terms of the load matrix I is nonuniform, the Knockout Switch behaves similarly under this traffic pattern as it does under uniform traffic because it is a nonblocking switch. That is because all the outputs receive the same amount of traffic. Actually, we can easily prove that as $N \to \infty$, (15) reduces to the Poisson process independent of the values of ρ_1 and ρ_2. We considered this traffic pattern since it has been known that the multistage packet switches such as banyan networks seriously degrade their performance under this traffic pattern [3]. This shows again the advantage of the Knockout Switch over banyan networks.

Traffic Pattern 2: Next we consider the following nonuniform traffic pattern. The load matrix I is partitioned as

$$I = [I_1(\rho_1) \quad I_2(\rho_2)].$$

Here the nonuniform load matrix I is partitioned at the center, and I_1 and I_2 are uniform. The row sum of I_1 is ρ_1 (e.g. heavy traffic) while the row sum of D_2 is ρ_2 (e.g., light traffic) such that $\rho_1 + \rho_2 = \rho$. In other words, there are two equal-sized destination groups with the probabilities of a packet originating at a source going to destination group1 and destination group 2 being ρ_1 and ρ_2, respectively. This may happen in reality as traffic to some destinations is heavier than the traffic to others.

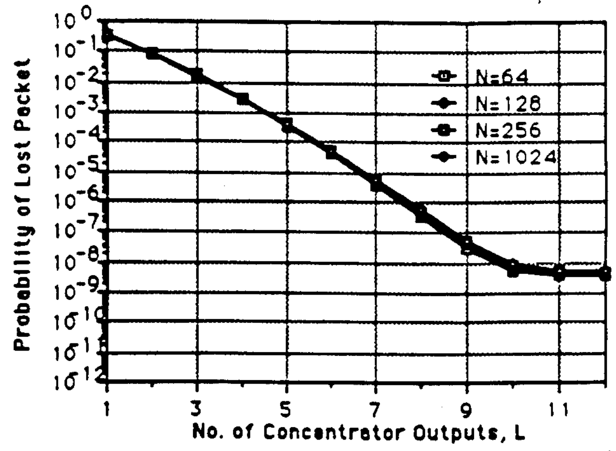


Fig. 17. Probability of lost packet versus threshold L: traffic pattern 2 (input load = 0.45, r = 1, m = 80).

Let $\rho_1 = r\rho$ and $\rho_2 = (1-r)\rho, 0.5 \leq r \leq 1.0$, and hi be an output port in destination group 1 (high traffic group). Then under this traffic pattern, q_{hi}^k is given by

$$q_{hi}^k = \binom{N}{k}\left(\frac{2r\rho}{N}\right)^k\left(1-\frac{2r\rho}{N}\right)^{N-k}. \tag{16}$$

Figs. 16–17 show, for $\rho = 0.45$, probability of packet loss versus L for various N's. To see the worst case, r was set to 1.0. It is seen that with $L = 8$ and $m = 80$, we can achieve a probability of packet loss less than 10^{-6} for a large N. Therefore, it is seen that the Knockout switch is stable under this traffic pattern, given sufficient buffers. In contrast, the performance of the multistage packet switches such as banyan networks degrades significantly under such a traffic pattern [3].

V. Conclusions

We have presented a new analytic model of the Knockout Switch which is more general than the earlier model [14]; it accepts any traffic patterns as an input, and considers the combined effects of a concentrator and a shared buffer on the lost packet probability. The accuracy of the new model has been verified by comparing the results obtained from the new model to the known results under uniform traffic.

With the new model, the performance of the Knockout Switch has been investigated under several nonuniform traffic patterns of interest. The Knockout Switch with 8 concentrator outputs and the buffer size of 40, an effective configuration for uniform traffic, has been shown stable under hot-spot traffic with the hot-spot rates which do not overload the hot-spot output.

Under the point-to-point traffic and traffic pattern2, it has been shown that the buffer size needs to be increased from 40 to 80 to keep the lost packet probabilities under 10^{-6}.

Acknowledgment

The authors would like to thank the anonymous reviewers for their constructive comments which were helpful for improving the paper.

References

[1] L. Bosack and C. Hedrick, "Problems in large LANs'," *IEEE Network*, vol. 2, pp. 49–56, Jan. 1988.

[2] K. Y. Eng, M. G. Hluchyj, and U. S. Yeh, "A Knockout Switch for variable-length packets," *IEEE J. Select. Areas Commun.*, vol. SAC-5, pp. 1426–1435, Dec. 1987.

[3] U. Garg and Y. P. Huang, "Decomposing banyan networks for performance analysis," *IEEE Trans. Comput.*, vol. C-37, no. 3, pp. 371–376, Mar. 1988.

[4] M. G. Hluchyj and M. J. Karol, "Queueing in space-division packet switching," *INFOCOM '88*, 1988, pp. 334–343.

[5] IEEE, "Special Issue on Broadband Communications," *IEEE J. Select. Areas Commun.*, vol. SAC-5, no. 8, Oct. 1987.

[6] Y. C. Jenq, "Performance analysis of a packet switch based on single-buffered banyan network," *IEEE J. Select. Areas Commun.*, vol. SAC-30, pp. 1014–1021, Dec. 1983.

[7] M. J. Karol and M. G. Hluchyj, "The Knockout Switch: Principles and performance," in *Proc. 12th Conf. on Local Comput. Networks*, 1987, pp. 16–22.

[8] M. J. Karol, M. G. Hluchyj, and S. P. Morgan, "Input versus output queuing on a space-division packet switch," *IEEE Trans. Commun.*, vol. COM-35, pp. 1347–1356, Dec. 1987.

[9] J. K. Kulzer and W. A. Montgomery, "Statistical switching architectures for future services," in *Proc. Int. Switching Symp.*, May 1984.

[10] P. W. Matthewson and S. R. Wilbur, "An integrated services switching system based upon a single-buffered banyan network," in *Proc. IEEE INFOCOM*, 1987, pp. 766–772.

[11] G. F. Pfister and V. A. Norton, "'Hot spot' contention and Combining in Multistage Interconnection Networks," *IEEE Trans. Comput.*, vol. C-34, pp. 943–948, Oct. 1985.

[12] J. S. Turner, "Design of an integrated services packet network," *IEEE J. Select. Areas Commun.*, vol. SAC-4, pp. 1373–1380, Nov. 1986.

[13] L. T. Wu, "Mixing traffic in a buffered banyan network," in *Proc. ACM 9th Data Commun. Symp.*, 1985, pp. 134–139.

[14] U. S. Yeh, M. G. Hluchyj, and A. S. Acampora, "The Knockout Switch: A simple, modular architecture for high-performance packet switching," *IEEE J. Select. Areas Commun.*, vol. SAC-5, pp. 1274–1283, Oct. 1987.

[15] H. Yoon, K. Y. Lee, and M. T. Liu, "Performance analysis of multibuffered packet-switching networks in multiprocessor systems," *IEEE Trans. Comput.*, vol. 39, pp. 319–327, Mar. 1990.

Hyunsoo Yoon received the B.S. degree in electronics engineering from the Seoul National University, Seoul, Korea, in 1979, the M.S. degree in computer science from the Korea Advanced Institute of Science and Technology, in 1981, and the Ph.D. degree in computer and information science from Ohio State University, Columbus, in 1988.

From 1978 to 1980, he was with the Tongyang Broadcasting Company, Korea. From 1980 to 1984 he was with the Computer Division of the Samsung Electronics Company, Korea, and from 1988 to 1989, he was with AT&T Bell Laboratories as Member of Technical Staff. He is currently an Associate Professor at the Korea Advanced Institute of Science and Technology. His main research interests include parallel computer architectures and communication protocols.

Ming T. (Mike) Liu (M'65–SM'82–F'83) received the B.S.E.E. degree from the National Cheng Kung University, Tainan, Taiwan, in 1957, and the M.S.E.E. and Ph.D. degrees from the Moore School of Electrical Engineering, University of Pennsylvania, Philadelphia, in 1961 and 1964, respectively.

Since 1969 he has been with the Ohio State University, Columbus, where he is currently Professor of Computer and Information Science. Since 1974, he has been actively involved in research and development of computer networking and distributed computing and has published over 100 technical papers in this and related areas.

Dr. Liu has received several awards for his technical contributions and for his dedicated services to the IEEE Computer Society. He was Program Co-Chair of the 1981 International Conference on Parallel Processing; Distinguished Visitor of the IEEE Computer Society from 1981 to 1984; Chairman of the Technical Committee on Distributed Processing from 1982 to 1984; Program and General Chairman of the Fifth and Sixth International Conference on Distributed Computing Systems, respectively; Chairman of the ACM/IEEE Eckert-Mauchly Award Committee from 1984 to 1985; Computer Society's Vice President for Membership and Information in 1986; and a member of the IEEE Fellow Committee from 1986 to 1988, among others. Since 1982, he has served as Guest Editor, Editor, and Editor-in-Chief of the IEEE Transactions on Computers. He has been elected three times as a member of the Computer Society's Governing Board from 1984 to 1990.

Kyungsook Y. Lee received the B.S. degree in chemistry from the Sogang University, Seoul, Korea, the M.S. degree in computer science from the University of Utah, and the Ph.D. degree in computer science from the University of Illinois at Urbana-Champaign.

She is and Associate Professor in the Department of Mathematics and Computer Science, University of Denver. Previously, she was on the faculty of the Department of Computer and Information Science, The Ohio State University. Her research interests include parallel computer architecture, parallel and distributed computing, communication networks, and optical networks.

Young Man Kim received the B.S. degree in mechanical engineering from the Seoul National University, Seoul, Korea, in 1980, and the M.S. degree in mechanical engineering from the Korea Advanced Institute of Science and Technology (KAIST), Seoul, Korea, in 1982. He also received the M.S. and Ph.D. degrees in computer science from the Ohio State University, Columbus in 1988 and 1992, respectively.

From 1983 to 1986 he was a system design engineer for various types of Factory Automation product at Gold Star Company, Seoul. Since 1992 he has worked as a researcher for Mitsubishi Material Corporation, Omiya City, Japan. His interests are in computer architecture, communication network, parallel processing, database system, and [illegible] processing

C. **C. H. Wu, L. S. Koh and M. T. Liu,**
"A Synchronization and Compensation Protocol for Multimedia Communication Systems,"
Proc. 1995 IEEE Int'l Conf. on Network Protocols,
pp. 252–259, November 1995.

A Synchronization and Compensation Protocol for Multimedia Communication Systems*

Chao-Hui Wu Liang-Seng Koh Ming T. Liu
Department of Computer and Information Science
The Ohio State University
Columbus, Ohio 43210-1277

Abstract

Media synchronization and freedom from starvation at destination devices are the two most important problems in a multimedia communication system. In order to solve these problems at the same time, this paper presents a protocol combining the scheduling scheme and the buffering scheme with underflow threshold (SBUF). In this protocol, by controlling the buffer sizes and modifying transmission schedule of the synchronizer, synchronization can be achieved without synchronizing clocks during a connection. In this paper, the algorithm of the protocol is described. The setting of the system parameters and network QOS to support the proposed protocol are also given in this paper. A simulation is performed for comparison of four synchronization schemes: scheduling schemes (with and without clock synchronization), marker scheme, and the proposed SBUF protocol. The results confirm that the proposed protocol guarantees satisfaction of QOS requirements and also performs better than the other schemes in terms of media synchronization and freedom from starvation at destination devices.

1 Introduction

A real-time multimedia application consists of different media types such as text, voice, image, video and audio. Data from various media that are generated at the same time need to be played back simultaneously or within an acceptable range at the destination. Due to the unique characteristics of each medium, different channels are used to carry different media at the same time. Consequently, different delays or jitters will be introduced into different media at the destination. Moreover, different media may go through different encoding/decoding procedures which have varying processing latencies. The frequencies of some physical devices may not match with others; therefore, the existence of different clock drifts between devices is very possible. See Figure 1 [1, 2]. The above factors lead to asynchrony among media streams during playback at the destination. An effective synchronization and compensation protocol is needed to prevent or remedy the possible asynchrony problems so as to guarantee the playback quality of multimedia applications.

Most work on multimedia synchronization protocols can be classified into two categories: scheduling schemes and marker schemes. In scheduling schemes [3, 4, 5], either a global network clock or a clock synchronization technique is assumed to be available. Each node or switch has a corresponding "transmission/display schedule." If all clocks in the system are synchronized, synchronization will be achieved if all devices transmit and play back data frames according to their respective schedules. However, in this scheme, the quality of the media synchronization highly depends on how fine clocks in the system can be synchronized. The current well-referenced clock synchronization technique, the Network Time Protocol (NTP) [6], can synchronize clocks within a range of +/- 5 ms. It still cannot satisfy some applications which need fine synchronization (such as down to 100 ms between media). In marker schemes [7], special control frames are added to mark points that need to be synchronized in the media streams. To ensure that synchronization is achieved among various media streams, a synchronizer realigns these control frames before data frames are played at the destination. However, the maker scheme cannot guarantee Quality of Service (QOS) when clock drift or data loss occurs. In both schemes, the buffering technique is adopted by using buffers to store early arrived data frames. Both the above schemes introduce extra overhead into the system. In some environments with very limited bandwidths, such as wireless communications, periodically synchronizing clocks across the system or inserting control frames can be very costly. Moreover, most of the protocols fail to compensate for the consequences of data loss and clock drift. As a result, the device starvation problem can still occur.

In this paper, a synchronization and compensation protocol is proposed to ensure media synchronization and freedom from starvation at destination devices. The protocol is a combination of the scheduling scheme and the buffering scheme with underflow threshold. The synchronization among media is achieved at the synchronizer by sending out all media frames according to the respective schedules at the synchronizer. To maintain synchronization at the destination, the buffer sizes of the destination devices are minimized to be three. The schedule of the synchronizer is modified to cope with the starvation problem caused by the clock drift between the synchronizer and the destination. Underflow threshold is defined and used to prevent overduplication of frames while compensating

*Research reported herein was supported by U.S. Army Research Office, under contract No. DAAL03-92-G-0184. The views, opinions, and/or findings contained in this paper are those of the authors and should not be construed as an official Department of the Army position, policy or decision.

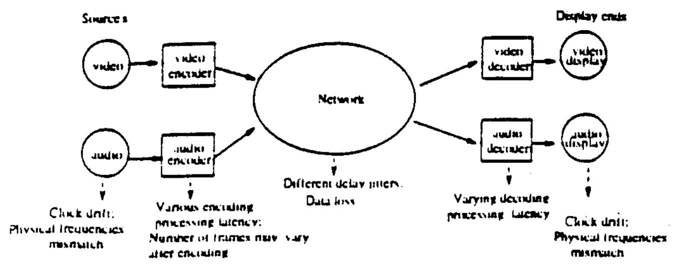

Sources
Display ends
video
video encoder
audio
audio encoder
Network
video decoder
video display
audio decoder
audio display
Clock drift: Physical frequencies mismatch
Various encoding processing latency: Number of frames may vary after encoding
Different delay jitters. Data loss
Varying decoding processing latency
Clock drift: Physical frequencies mismatch

Figure 1: Asynchrony factors of a multimedia system

for the starvation problem. The clock synchronization is not needed during a connection. The assumption of the proposed protocol is that the upper bound of the drift between clocks is known. The simulation performed in this paper shows that the proposed protocol guarantees satisfaction of all QOS requirements. In addition, the proposed protocol is the best of the four schemes in eliminating the destination devices starvation and synchronization problems.

The rest of this paper is organized as follows. In Section 2, the asynchrony problems are described, and the QOS requirements used to prevent these asynchrony problems are also specified. In Section 3, the proposed protocol is introduced. The algorithm, the setting of system parameters and the QOS translation between the application and the network are also given. A simulation study comparing four schemes is given in Section 4. Finally, an extension of this protocol is examined and concluding remarks are presented in Section 5.

2 QOS Requirements

Before we present our protocol, the asynchrony problems that this paper is going to address and QOS parameters to be used later are described in this section.

2.1 Asynchrony problems

The goal of a synchronization and compensation protocol is to remedy the asynchrony problems at destinations such that the quality of an application can be maintained. It is necessary to identify every possible consequence and problem caused by asynchrony factors so that the proposed protocol can be designed to prevent and compensate for these undesired asynchrony problems completely. The possible undesired consequences and problems are stated as follows.

Lip-sync problem: The image of a speaker's lips does not match his voice. If the supposedly synchronized picture and voice data arrive at the destinations within the range 70-150 ms, human viewers cannot detect the asynchrony between them.

Destination devices starvation problem: A destination device has no data frame to play when the scheduling deadline is due. The human visual and auditory systems can perceive the remaining effect of a picture and voice for 30-40 ms. Thus, display devices need to playback data periodically at fixed time intervals of 30-40 ms in order to preserve the effect of continuity in the picture and voice.

Duplication gap problem: The duplication gap problem occurs when the same frame is played too many times, so that human viewers are able to detect the abnormal result. It is a result of overduplication by a compensation protocol for lost or late arrival data.

Media skip problem: The media skip problem occurs when two non-consecutive data frames are played consecutively at the destination. Data loss during transmission and data dropped by the synchronization protocol are the main causes of this problem. This consequence cannot be compensated because retransmission is not possible in applications that have real-time constraints. Hence, when a synchronization protocol is designed, it is important not to introduce too much dropping of data frames while trying to overcome the other problems.

The above undesirable problems can be prevented if appropriate QOS requirements are specified in an application and these QOS requirement are maintained during a connection by a designed protocol and the underlying network. Therefore, in the following two subsections, we first identify the QOS requirements to be specified by the application for addressing the problems and then present the network support needed in terms of network QOS requirements. Based on these two sets of QOS parameters, an efficient protocol is proposed to address these problems in Section 3. The translation between two sets of QOSs is shown in Section 3.4.

2.2 QOS requirements specified by an application

In order to eliminate the undesirable problems described above, we define a set of QOS requirements specified by an application in terms of the following parameters:

- **transmission rate (frames/sec) $\hat{\Omega}_t$, a.k.a. sampling rate:**
  the number of data frames transmitted/displayed per second. This parameter is used to address the destination devices starvation problem.
- **maximum end-to-end delay allowed (sec) $\hat{D}_t$:**
  the maximum allowed time interval for a data frame between generating time at the source and displaying time at the destination. This is used to address the real-time constraint for an application. It can also help to prevent the destination devices starvation problem.
- **maximum out-of-sync allowed range (sec) $\hat{\Phi}_t$:**
  the maximum allowed out of sync range which can be tolerated between media. If this requirement can be guaranteed, the lip-sync problem can be eliminated.
- **maximum skip frames allowed at one time (frames) $\hat{K}_t$:**

the maximum number of consecutive frames that can be skipped. This parameter is for preventing media skip problem.

- **maximum duplication allowed per frame $\hat{U}_i$:**
  the maximum number of duplications for a frame. This parameter is needed for avoiding duplication gap problem.

When service is requested, an application needs to specify a set of values to these QOS parameters. This set of values defines the QOS requirements for the application. For an application, if all its QOS requirements can be satisfied, undesirable asynchrony problems will be avoided at the destination.

2.3 QOS parameters supported by the network

The network support needed are specified in terms of the network QOS parameters. The network QOS parameters include:

- **data transmission rate (frames/sec) ω_i**
  the number of frames transmitted per second by the network
- **data loss rate p_i**
  the percentage of data allowed to be lost by the network
- **maximum network delay allowed d_i**
  the maximum allowed end-to-end network delay
- **network jitter j_i**
  the maximum delay variance allowed in the network

The values of these network QOS parameters are derived from the application QOS requirements plus the possible overhead introduced by the synchronization and compensation protocol. The relationship between the network QOS requirements and the others is shown in Section 3.4.

3 Protocol

In this section, a synchronization and compensation protocol is proposed to solve the lip-sync and starvation problems and to avoid the duplication gap and media skip problems. First, the compatibility of the proposed protocol with the current existing network models is discussed. Then, the protocol is described in details to show how each asynchrony problem is prevented. Based on the proposed protocol, how the system parameters are tuned to support the QOS requirements is also shown in this section. Finally, the transformation between the network QOS parameters and the other parameters are presented.

3.1 Network model

In existing networks, since there are no corresponding functions to address synchronization and its associated problems, the proposed synchronization protocol can be added as a layer to an existing network model. In order to absorb all the asynchrony factors, the synchronization layer should be placed as

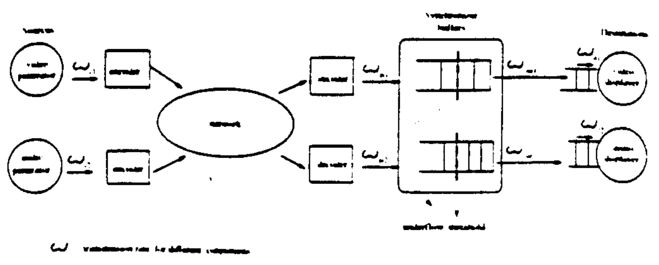


Figure 2: Network architecture for a multimedia system

close as possible to the application layer. For example, in order to absorb various latencies of decoding procedures for different media, a synchronization layer should be placed between the application layer and the presentation layer in the OSI model, and between the ATM adaptation layer and the application layer in the ATM model.

A multimedia communication system consists of several components. The proposed protocol requires a synchronizer near the destination to coordinate synchronization among media streams, as shown in Figure 2. It is assumed that the new data will overwrite the most recently written data frame in the buffer if the buffer is full when new data frames arrive.

Data frames are generated periodically at the sources according to the sampling/display rate of the devices. A time stamp or sequence number is stamped on each data frame, so that the expiration time can be checked at the destination. The network transmits data frames according to the network QOS negotiated when the connection was set up. When data frames arrive at the synchronizer, they will first be stored in the associated buffer. Data frames will be sent to the destination according to the transmission schedule in the synchronizer. At the destination, newly arrived data frames are stored in the buffers of the destination devices before being played back. A device periodically retrieves data from its buffer and plays it back according to the playback (retrieval) rate of the device.

3.2 Protocol description

The synchronization protocol proposed in this paper uses both the scheduling technique and the buffering technique. Basically, asynchrony factors and effects originating from the processes before the synchronizer can be detected and compensated for by the synchronizer. For example, the length of a buffer in the synchronizer indicates the possible clock drift relationship between the synchronizer and the sources. If the length of the data frames in the buffers keeps increasing, it implies that the transmitting speed of the sources may be faster than that of the synchronizer, or vice versa. Using scheduling and buffering techniques, the synchronizer smoothes the skew of the arrival time of data streams and makes sure the outgoing data frames are synchronized or within the out-of-sync range. The lost data or late arrival data will be compensated for by duplicating data frames.

However, asynchrony effects that occur after the synchronizer but before playback are hard to control. The clock drift relationship between the synchronization and destination de-

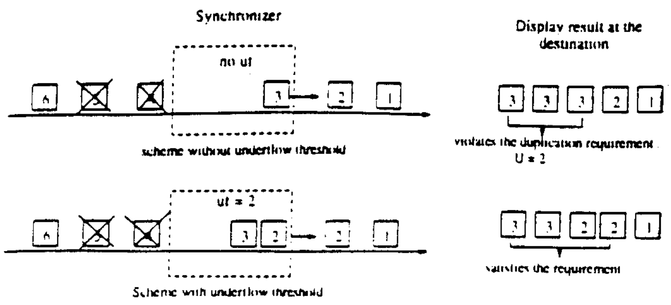

Synchronizer
no uf
6
3
2
1
scheme without underflow threshold
uf = 2
6
3
2
2
1
Scheme with underflow threshold
Display result at the destination
3
3
3
2
1
violates the duplication requirement
U = 2
3
3
2
2
1
satisfies the requirement

Figure 3: Underflow threshold smoothes the duplication gap

vices cannot be known if there is no feedback from the destination devices. Device starvation may occur if the clock at the destination is faster than the one at the synchronizer. The lip-sync problem may occur if the clock drifts between the destination devices are different, despite media streams being synchronized at the synchronizer. The way the proposed protocol overcomes these two problems is described separately in the sections that follow.

3.2.1 Solving the starvation problem

The device starvation problem occurs either when data frames are lost during transmission or when the clock at the destination is faster than the clock at the synchronizer. In the first case, the synchronizer uses an *underflow threshold (uf)* to indicate a potential insufficient data situation. When the number of data frames in the buffer drops to the level of the uf threshold, the synchronizer will start making copies of the outgoing data frames so that the duplication per frame can be minimized without exceeding the duplication requirement. For example, as shown in Figure 3, a scheme without the underflow threshold does not save enough frames in the buffer. If the data frames with sequence numbers 4 and 5 are lost during transmission, the scheme without uf can only compensate for the lost data by repetitively transmitting a copy of the previous frame. As a result, the duplication per frame at the destination may exceed the QOS requirement specified by the application of a maximum duplication rate of $U = 2$. However, in the scheme using uf threshold, the duplication gap can be smoothed by always saving uf frames in the buffer.

In the second case, starvation is prevented by modifying the transmission rate of the synchronizer so as to guarantee that the destination playback rate is not faster than the synchronizer transmission rate no matter how bad the clock drift is. That is, the schedule for the synchronizer is modified to always send out more frames as if the clock at the destination were the fastest and the clock at the synchronizer were the slowest. Assuming the clock drift is bounded, the bound is defined as B per second. In order to prevent the starvation problem, the synchronizer assumes itself to have the slowest clock rate, and the destinations to have the fastest clock rate. If the original scheduled transmission rate for the synchronizer is Ω frames/second, it will be modified to a rate of at least $\Omega(1 + B)$ to match the fastest possible transmission rate at the destination. In this way, the transmission rate of the synchronizer will always be the fastest or one of the fastest of all the components. As a result, starvation at the destination will never occur. With the aid of uf threshold, data frames can be properly duplicated so that sufficient data is supplied for the synchronizer to transmit.

Using this method, though a duplication problem is created, it can be solved easily, thus avoiding the starvation problem at the destination. At the same time, by applying uf threshold, the duplication per frame is kept under control. If the clock drift between the destination and the synchronizer is not as predicted, extra frames transmitted by the synchronizer will be overwritten by new arrival data frames that are newly arriving at the destination buffers.

3.2.2 Solving the lip-sync problem

The size of the buffers at the destination site will affect the synchronization quality of the multimedia system. If clock drifts among the destination devices differ, their real data display rates will be different from what they declare. Despite all data frames being synchronized at the synchronizer, a destination device with a faster clock will display more data frames than a destination device with a slower one. The queue for the slower device's buffer will gradually increase. This out-of-sync effect among media may occur after a period of time. To avoid this situation, and to consume the extra data frames sent by the synchronizer, the buffer size should be as small as possible so that newly arriving data frames can be played back right away and the synchronization achieved at the synchronizer can be maintained at the destination. The shorter the period of time data frames wait at the buffers, the less the chance that they will be out of synchronization.

However, the read-write conflict at the destination buffer can be avoided only if the buffer size is large enough to contain at least three data frames. This is illustrated via counterexamples. Note that writing by the synchronizer is more frequent than reading by the destination devices. This is because, after modification of the schedule, the synchronizer has the fastest transmission rate of all components. When the buffer is full, the newly arriving data frame will overwrite the data frame the synchronizer wrote just before it; otherwise, the new data frame will be written in an empty buffer. If the buffer size is one, then while the display device is reading, the newly arriving frame may rewrite it, and a read-write conflict occurs. Possible effects may be that the display frame proves incomplete or the display of a frame is interrupted. If the buffer size is two, as shown in Figure 4, the read-write conflict may occur when the destination device finishes reading the first buffer and starts to read the second buffer. If the buffer size is three, there is always one more buffer that no read or write will access. Whichever agent finishes first can move to that buffer. Therefore, the read-write conflict can be avoided.

In summary, to avoid possible asynchrony at the destination, the smaller buffer size of the destination should be as small as possible. However, as discussed above, the minimum size needed to avoid a read-write conflict is at least three. Therefore, the optimal buffer size for the destination device should be three.

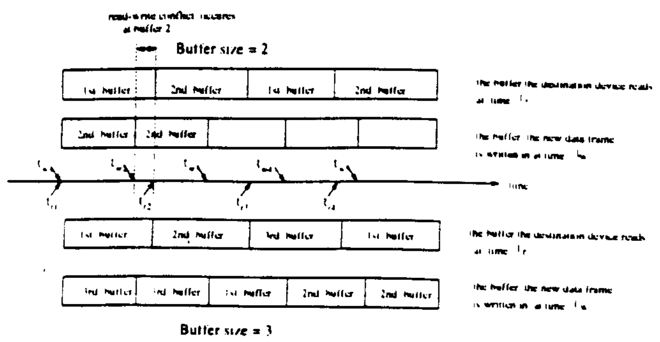

read-write conflict occures at buffer 2
Buffer size = 2
1st buffer
2nd buffer
1st buffer
2nd buffer
the buffer the destination device reads at time t_r
2nd buffer
2nd buffer
the buffer the new data frame is written in at time t_w
time
1st buffer
2nd buffer
3rd buffer
1st buffer
the buffer the destination device reads at time t_r
3rd buffer
3rd buffer
1st buffer
2nd buffer
2nd buffer
the buffer the new data frame is written in at time t_w
Buffer size = 3

Figure 4: Examples of read-write conflict

3.3 Algorithm and system parameters setting

To sum up the protocol described above, an algorithm of the proposed protocol for the synchronizer is given in this section. Whenever needed, system parameters will be set to support the QOS of the application.

The system parameters used in the *synchronizer* are as follows. See also Figure 2.

- ω_{out}: new transmission rate after modification
- uf: underflow threshold to indicate any possible insufficient data situation
- *counter*: the maximum number of duplications per frame
- buf: the buffer size in the synchronizer

When the connection is set up, the clocks in the system are synchronized, and each component is given a schedule, which normally is the transmission rate/sampling rate of the application. The transmission schedule for the synchronizer is modified to match the possible clock drift between the synchronizer and the destination so that the starvation problem can be avoided. When a new deadline is up, if the number of data frames in the buffer is more than or equal to *uf* threshold, the synchronizer retrieves a data frame from the associated buffer and sends it out. However, if the number of data frames in the buffer is less than the uf threshold, and if the data frame at the head of the associated buffer has not been duplicated more than *counter* times, the synchronizer sends out a copy of that data frame. The detailed algorithm is shown in Figure 5.

In order to support the proposed protocol, system parameters need to be tuned accordingly. The setting of system parameters is explained further in the following subsections.

3.3.1 Setting a new transmission rate ω_{out}

It is assumed that clock drift is bounded at B per second between a clock and a precision timing source (calibrated cesium clock.) When a clock is operating as precious as a calibrated cesium clock, the transmission rate of a media stream i for each component in the system is Ω, as specified in the QOS parameters. However, because of local clock drift, the real transmission rates of components may not be Ω as compared to the precision timing source. The starvation problem may occur if the clock at the destination is faster than the one at the synchronizer. In the worst case, the real display rate of a display device would be $\Omega(1+B)$, and the real transmission rate of the synchronizer would be $\Omega(1-B)$, according to a precision timing source. In the worse case, to match the transmission rate of the destination the synchronizer must be x times faster than its real rate. That is,

$$x * \Omega(1-B) = \Omega(1+B)$$

The new transmission ω_{out} rate for the synchronizer is set to:

$$\omega_{out} = x * \omega_{syn} = \frac{1+B}{1-B}\omega_{syn} \qquad (1)$$

where ω_{syn} is the original transmission rate of the synchronizer as compared to a precision timing source.

3.3.2 Setting the *counter*

In order to guarantee QOS requirements regarding the maximum duplications per frame at the destination, the maximum number of duplications per frame (*counter*) at the synchronizer is set to be U, the maximum allowable duplications per frame specified by the application.

$$counter = U \qquad (2)$$

Since the number of copies of each frame transmitted from the synchronizer is less than or equal to U and since the end device will not duplicate data frames, the maximum number of copies of a frame displayed at the destination will be less than or equal to U. Thus the situation of overduplication of frames at the destination can be eliminated.

```
Modify the schedule of the synchronizer
For every media stream $i$:
  $counter_i = U_i$
  Set the value of $uf_i$ based on (5)
For every media stream $i$:
  while a deadline on the new schedule is due
    if $0 <$ the queue length of the $buffer_i < uf_i$ then
      if $counter_i > 0$ then
        synchronizer sends out a copy of the first data frame
        in the $buffer_i$
        $counter_i = counter_i - 1$
      else
        synchronizer sends out the first data frame in $buffer_i$
        $counter_i = U_i$
    else if the queue length of the $buffer_i \geq uf_i$ then
      synchronizer sends out the first data frame in its buffer
      $counter_i = U_i$
end
```

Figure 5: The algorithm for the synchronizer

However, if duplicating U copies of every data frame generated at the source still cannot meet the display rate of the destination, more than U copies per frame are necessary to achieve freedom from starvation at the destination. Thus, the proposed protocol can only guarantee freedom from the overduplication problem if the clock drift bound, B, and the duplication, U, are related as shown below.

$$\omega_{source} * U > \omega_{destination} \tag{3}$$

The data generation rate at the source is expressed as ω_{source}; $\omega_{destination}$ is the data playback rate at the destination (indicated as ω_s, and ω_d, in Figure 2). In the worst case, $\omega_{source} = (1-B)\Omega$ and $\omega_{destination} = (1+B)\Omega$. Therefore, according to inequality (3), it can be derived as:

$$U \geq \frac{1+B}{1-B} \tag{4}$$

Fortunately, the bound of clock drift B is hardly likely to be more than 10^{-1} per second. In that case, according to (4), U as specified by the application will be at least 1.3, which is generally acceptable for most of the applications. Therefore, in most cases, the proposed protocol can satisfy the overduplication constraints of QOS requirements.

3.3.3 Setting the uf

Suppose the the data loss rate of the network is p, which is also the probability that a data frame will be lost. Denote $jitter$ as network jitter and ω_{in} as the data frames arrival rate from the network. Then uf should contain sufficient data frames to make it certain that a new data frame will arrive before all the data frames in the associated buffer are gone. That is, the following inequality should hold:

$$\frac{uf * U}{\omega_{out}} \geq (\frac{1}{\omega_{in}} + jitter) * ((1-p) + 2p(1-p) + 3p^2(1-p) + \ldots)$$

Since $(1-p) + 2p(1-p) + 3p^2(1-p) + 4p^3(1-p) + \ldots \rightarrow \frac{1}{1-p}$, it can be rewritten as:

$$uf \geq \frac{(\frac{1}{\omega_{in}} + jitter) * \omega_{out}}{counter * (1-p)}$$

In the worst case, $\omega_{in} = (1-B)\Omega$ and $\omega_{out} = \frac{1+B}{1-B}(1+B)\Omega$, thus

$$uf \geq \frac{(1+B)^2(\frac{1}{1-B} + jitter * \Omega)}{(1-B)(1-p)U} \tag{5}$$

3.3.4 Setting the buffer size buf

The buffer overflow problem will not occur with a synchronizer because ω_{out} is always greater than ω_{in}. However, the buffer size (buf) at least should be larger than the uf in order to hold the data frames needed to be duplicated. Therefore, the buffer size buf can be set as

$$buf = uf + 1 \tag{6}$$

3.4 Setting the corresponding network QOS parameters

In order to support the QOS requirements of the application, the QOS parameters of the network are set as follows :

- **data loss rate (p):** The allowed data loss rate is mapped from the maximum skip allowed parameter (K) specified by the multimedia application. Data loss is not possible in the synchronizer buffers since $\omega_{out} \geq \omega_{in}$. It will occur only if data frames are overwritten by newly arriving data. Thus, p needs to satisfy the inequality (5).

- **data transmission rate (ω):** If possible, the data transmission rate of the network should be set as $\Omega(1+B)$, which is the fastest possible transmission rate at the sources. However, it is very rare that clocks at both sources and destinations will drift so much. In order to achieve better network utilization, the data transmission rate is set as the scheduled transmission rate Ω. The simulation result confirms that it is sufficient to guarantee the application QOS satisfaction.

- **end-to-end delay (d):** The time spent on the synchronizer buffer will be $\frac{buf-uf}{\omega_{out}} + \frac{uf*c}{\omega_{out}} + \frac{3}{\omega_{destination}}$. Therefore, the end-to-end delay supported by the network should be

$$d \leq D - (\frac{buf-uf}{\omega_{out}} + \frac{uf * counter}{\omega_{out}} + \frac{3}{\omega_{destination}})$$

  In the worst case, ω_{out} and $\omega_{destination}$ are both slowest; that is, $\omega_{out} = \frac{1+B}{1-B}(1-B)\Omega = (1+B)\Omega$ (by (1)) and $\omega_{destination} = (1-B)\Omega$. Together with (2) and (5),

$$d \leq D - (\frac{1 + uf * U}{(1+B)\Omega} + \frac{3}{(1-B)\Omega})$$

- **network jitter ($jitter$):** The network jitter should satisfy the inequality (5).

4 Simulation Results

In this section, a simulation is carried out to compare the performance of the proposed protocol with other synchronization schemes, namely, the marker scheme, the scheduling scheme with synchronized clocks and the plain scheduling scheme without synchronized clocks. Four schemes are implemented as follows:

1. **Marker scheme (MK):**
   As described in Section 1, in the marker scheme special control frames are inserted into the media stream periodically. These special control frames are then realigned at the synchronizer. In this simulation, controlled frames are inserted every 3.3 seconds, or approximately every 100 frames for the video application.

2. **Scheduling scheme with synchronized clocks (SSC):**
   Data frames are transmitted according to their schedules and their local clocks. In the system, all clocks are synchronized within -/+ 5 ms.

The QOS requirements specified by an application[4, 8]:

media	$\hat{\Omega}$ frms/s	$\hat{D}$ ms	$\hat{\Phi}$ ms	$\hat{U}$ per frm	$\hat{K}$ frms
video	30	250	100	2	2
voice	3000	250	100	2	2

The QOS parameters supported by the network:

media	ω frames/sec	d ms	j ms	p
video	30	220	10	10^{-3}
voice	3000	220	10	10^{-1}

Figure 6: QOS parameters used in simulation

3. **Plain scheduling scheme without synchronized clocks (PS):**
   This scheme is similar to the SSC but clocks in the system are not synchronized.

4. **Scheduling scheme with underflow threshold (SBUF):**
   This is the protocol proposed in this paper. Parameters are set up as described in Section 3.

Given the same QOS requirements from an application and the same network support, the percentage of the QOS requirements violations between schemes is compared. The application selected combines video and audio media. The QOS requirements and the parameters supported by the network are listed in Figure 6 and Section 2.

The performance parameters investigated are :

out-of-sync rate (%): the percentage of out-of-sync frames as compared to the total frames generated

overduplication rate (%): the percentage of overduplicated frame as compared to the total frames generated

over skip rate (%): the percentage of frames where the skip range is more than that allowed by the QOS parameter as compared to the total frames generated

starvation rate (%): the percentage of destination starvation as compared to total frames generated

average max out-of-sync (ms): the average value of the maximum out-of-sync occurrences displayed at the destination

duplication (%): the percentage of duplicated frames as compared to the total frames generated

skip (%): the percentage of frames skipped as compared to the total frames generated

The first four parameters are indicators of QOS violations. They should all be zeroes in order to satisfy the QOS requirements specified by the application. The last three parameters are quality indices which show the quality of service for the application. The smaller their values are, the better the quality is. The simulation is carried out under different clock drift bounds (B) and under different clock drift situations among the sources, synchronizers and the destination devices. The results are shown in Figure 7.

B = 5 ms, U=2, and K=2					
Scheme	out of sync (%)	starv. (%)	max out-sync (ms)	dupl. (%)	skip (%)
SSC	10.52	0.4	105.3	0.32	0.32
PS	10.52	0.4	105.3	0.32	0.32
MK	0	0.54	40.49	0	0.29
SBUF	0	0	46.47	1.35	1.23

B = 10 ms, U=2, and K=2					
Scheme	out of sync (%)	starv. (%)	max out-sync (ms)	dupl. (%)	skip (%)
SSC	10.52	0.4	105.3	0.32	0.32
PS	14.93	0.61	158.9	0.53	0.47
MK	0	0.71	44.02	0	0.41
SBUF	0	0	51.25	2.28	2.25

B = 100 ms, U=2, and K=2					
Scheme	out of sync (%)	starv. (%)	max out-sync (ms)	dupl. (%)	skip (%)
SSC	10.52	0.4	105.3	0.32	0.32
PS	40.00	6.58	1620.46	3.97	3.85
MK	0	5.90	47.90	0	4.04
SBUF	0	0	52.37	15.43	15.3

Figure 7: Simulation results

From the results, the average over duplication rate and the average over skip rate are all zeroes in all four schemes. This means that four schemes will not introduce the duplication gap and media skip problems while compensating the other problems. However, only the proposed protocol, SBUF, has zero in the both the out-of-syn and starvation rates measurement. That is, only SBUF successfully remedies the out-of-sync and starvation problems. Besides, the value of the maximum out-of-sync parameter also indicates that the proposed protocol can achieve fine synchronization. Two of the quality indices, the duplication rate and the skip rate, are higher as compared to the other schemes. This is because the synchronizer creates extra duplication and transmission to prevent destination starvation from occurring. However, as long as duplication and skip are within the ranges of QOS requirements, the results are acceptable.

The marker scheme (MK) performs well except on the freedom from starvation requirement. The performance of the scheduling scheme with synchronized clocks (SSC) is stable. However, starvation and out-of-sync problems may still occur even when clocks are synchronized within +/- 5 ms. As expected, the performance of the plain scheduling scheme without synchronized clocks (PSS) is the worst of the four schemes. When clock drifts between clocks in the system increase, the quality of the application decreases.

The data we collected is based on a three minutes long application. If the duration lasts longer, the performance parameters values with percentages in the Figure 7 will remains the same. However, the average of the maximum out-of-sync (ms) will increase proportionally to the duration length.

The results of the out-of-sync rate and the starvation rate are also plotted in the Figure 8 and Figure 9, respectively. In Fig-

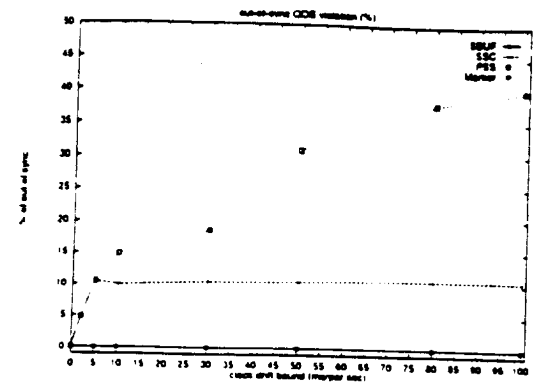


Figure 8: Comparisons in terms of out-of-sync rate

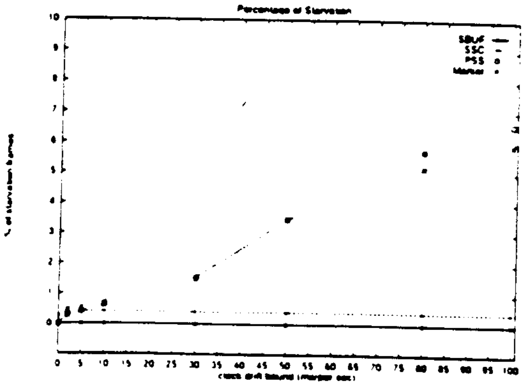


Figure 9: Comparisons in terms of starvation rate

ure 8, the marker scheme and the proposed protocol (SBUF) maintain no out-of-sync violation. The out-of-sync frame rate of SSC becomes stable at approximate 10% when the clock drift bound exceeds 5 ms since all clocks are synchronized within +/- 5 ms. For the PSS scheme, its out-of-sync rate is almost proportional to the clock drift bound. In Figure 9, the proposed SBUF protocol achieves starvation-free under any clock drift bounds. The SSC remains stable at 0.4%. The starvation rates for the marker scheme and the PSS scheme both increase proportionally to the clock drift bound. When the clock drift bound is less than 50 ms, the PSS has a slightly less starvation rate than the marker scheme. However, when the clock drift bound exceeds 50 ms, the marker scheme has smaller starvation rates comparing with the PSS.

5 Conclusion

In this paper, a protocol is proposed to simultaneously solve the synchronization and destination starvation problems. Clock synchronization is not needed during connection in this protocol. In this protocol, the synchronization is achieved at the synchronizer by transmitting data according to the respective schedules of the synchronizer. Synchronization is maintained at the destination by minimizing the buffer sizes of the destination devices to be three. The schedule of the synchronizer is modified to cope with the possible starvation problem. Underflow threshold is defined and used in the buffer of a synchronizer to prevent the possible violation of QOS requirements. The corresponding system parameters and the QOS parameters of the network are stated and set so as to support the proposed protocol. A simulation is done for comparing the four synchronization schemes. The results confirm that the proposed protocol guarantees the satisfaction of QOS requirements and also show better performance than the other schemes in terms of media synchronization and freedom from starvation at destination devices. Although the proposed protocol has a higher duplication rate and skip rate than the other schemes, the results are acceptable as long as they satisfy the QOS requirements.

This protocol could be further improved if the devices at the destination were given a feedback function. Only one feedback from each destination device would be needed. Based on the feedback, the synchronizer could accurately adjust its schedule to meet the display rates of the destination devices. The duplication percentage and skip percentage could be well controlled and it would be expected that they would decrease a great deal. The quality of the service would be improved dramatically.

References

[1] B. Furht, "Multimedia systems: An overview," *IEEE Multimedia Magazine*, pp. 47–59, Spring 1994.

[2] S. Browne, "Communication and synchronization issues in distributed multimedia database systems," in *Advanced Database Systems, Springer-Verlag*, pp. 381–396, 1993.

[3] T. D. Little and A. Ghafoor, "Multimedia synchronization protocols for broadband integrated services," *IEEE Journal on Selected Areas in Communications*, vol. 9, pp. 1368–1382, December 1991.

[4] J. Escobar, C. Partridge, and D. Deutsch, "Flow synchronization protocol," *IEEE/ACM Transactions on Networking*, vol. 2, pp. 111–121, Aprial 1994.

[5] M. Woo, N. U. Qazi, and A. Ghafoor, "A synchronization framework for communication of pre-orchestrated multimedia information," *IEEE Network*, vol. 1, pp. 52–61, January/February 1994.

[6] D. L. Mills, "Precision synchronization of computer network clocks," *Computer Communication Review*, vol. 24, pp. 28–43, April 1994.

[7] C.-J. Wang, L.-S. Koh, C.-H. Wu, and M.-T. Liu, "A multimedia synchronization protocol for ATM networks," in *Proc. IEEE International Conference on Distributed Computing Systems*, (Poznan, Poland), June 1994.

[8] A. Campbell, G. Conlson, F. Garlia, D. Hutchison, and H. Leopold, "Integrated quality of service for multimedia communications," in *Proc. IEEE INFOCOM '93*, pp. 732–739, 1993.

D. Y. Yang, T. H. Lai and M. T. Liu,
"Mobile Real-Time Communications in FDDI Networks,"
Proc. 1995 IEEE Int'l Conf. on Network Protocols,
pp. 201–208, November 1995.

Mobile Real-Time Communications in FDDI Networks*

Yibin Yang Ten-Hwang Lai Ming-Tsan Liu

Department of Computer and Information Science
The Ohio State University
Columbus, Ohio 43210-1277
E-mail: {yyang, lai, liu}@cis.ohio-state.edu

Abstract

We propose an architecture of FDDI-based mobile networks and address issues that arise in providing real-time communication services on such networks. A wide range of problems concerning synchronous bandwidth management and quality of service guarantee are identified. To solve these problems, we present a dynamic bandwidth management scheme, a source handoff protocol and two approaches to handling destination handoffs. These schemes make handoffs transparent to mobile users; no degradation in quality of service will be observed during handoffs. The proposed solutions are compatible with the FDDI standards.

1 Introduction

Wireless information networks have recently become a topic of intense interests. These networks are intended to provide mobile users with access to network resources and services anywhere at any time. An architecture of wireless networks, *cellular packet switch*, is proposed in [7]. It is based on the micro-cell structure and uses DQDB metropolitan area networks as infrastructure to connect wireless interfaces. Network control functions, such as call processing, mobility management and wireless resource management, are distributed among several interfaces in DQDB networks.

Our research is in a different direction. It is about extending existing FDDI networks with wireless terminals. FDDI, standing for fiber distributed data interface, is one of the most popular metropolitan area networks. It has been widely deployed. According to [9], the growth rate of FDDI networks was 80% in 1993 and close to 100% in 1994. We believe that the FDDI will continue to be popular and a mobile network based on it will be of great market value.

An ideal mobile system should provide mobile users with all services available to static (non-mobile) users. One of such services, real-time communication service [6, 11], is of particular importance in today's high speed networks. It can be used to support many applications such as manufacturing systems management, remote monitoring, robotic control, voice and video transmission. In [1, 14], the synchronous transmission capacity of FDDI networks is shown to be capable of supporting real-time communication service. It is desirable to extend this service to mobile users. One important application of this extension is in robotic communications. Communications between robots multiply their capabilities and effectiveness [4]. In a manufacturing environment, by providing mobile real-time communications between autonomous robots, the productivity is expected to increase. However, supporting real-time communications in a mobile network is a non-trivial task because a mobile user may move around. To cope with this situation, we first introduce a new architecture of mobile networks, then identify problems to support real-time communications in this architecture, and finally, design protocols and schemes to overcome these problems.

The rest of this paper is organized as follows. Section 2 is a preliminary of FDDI networks and real-time communications. Section 3 presents an architecture of FDDI-based mobile networks and discusses problems of real-time communications in this architecture. A dynamic bandwidth management scheme is proposed in Section 4. Solutions to problems caused by the mobile source and destination of a real-time connection are given in Sections 5 and 6, respectively. Finally, some conclusions are drawn in Section 7.

2 FDDI Networks and Real-Time Communications

In this section, we first briefly review the medium access control protocol (MAC) and the station management protocol (SMT) of FDDI networks. Then we introduce the concept of real-time channel and its extension to FDDI networks.

An FDDI network consists of a number of stations connected as a ring. It employs a *timed token MAC protocol*. A special control packet, *token*, circulates around the ring. Only the station possessing the token can transmit data. Data are classified into two categories: *synchronous* and *asynchronous*. A station transmits data according to the following rules: (1) each station is allocated a certain amount of bandwidth for transmitting synchronous data, h_i denoting the bandwidth allocated to station i;

*Research reported herein was supported in part by the Performance Analysis Lab of The Ohio State University and by U.S. Army Research Office, under contract No. DAAL03-92-G-0184. The views, opinions, and/or findings contained in this paper are those of the authors and should not be construed as an official Department of the Army position, policy or decision.

(2) each time a station i receives the token, it is allowed to send synchronous data for up to h_i units of time; (3) if the token arrives early, the station is allowed to send asynchronous data for an amount of time that equals the earliness of the token arrival; otherwise, it is not allowed to send asynchronous data. The readers are referred to [2, 9] for details.

There are some important time properties for the FDDI MAC protocol. During the ring initialization, all stations agree on a value called the target token rotation time (TTRT). Let Θ be the sum of latencies between the stations and Δ denote the time to transmit a maximum-size asynchronous message. It has been shown in [10] that if $\sum_i h_i \leq TTRT - \Theta - \Delta$, each station is guaranteed to see the token at least once in every $2TTRT$ time. This result is generalized in [5] to the following theorem:

Theorem 1 *If $\sum_i h_i \leq TTRT - \Theta - \Delta$, the time elapsed between any n consecutive token visits to station i is bounded by $nTTRT - h_i$.*

According to the station management protocol [3], the bandwidth management in FDDI networks is centralized and static. There is at least one synchronous bandwidth management process (BMP) in an FDDI network. For each FDDI station, there is a station management module. The bandwidth allocation is through a request/response frame exchange between the BMP and the SMT module of a station. When a station wants to allocate or release some synchronous bandwidth, it sends a resource allocation frame (RAF) request to the BMP. Upon receiving this request, the BMP sends back a RAF response indicating the success or failure of the request. For the bandwidth allocation, a station cannot increase its amount of bandwidth until a positive response is received from the BMP.

The concept of real-time channel was first proposed in [6]. It is a simplex virtual connection with the quality of service (QOS) guarantee. This concept is further extended in [14] to FDDI networks by the following definition.

Definition 1 A *real-time channel* in an FDDI network is characterized by a 5-tuple, $RC = (T, C, s, d, h)$, where s is a source station generating a sequence of packets; T is a lower bound on the interval between any two consecutive packets; C is the maximum packet length, measured by transmission time; d is the maximum delay on each packet (a packet must be transmitted in d units of time after its generation): and h is the amount of bandwidth allocated to s.

A real-time channel $RC = (T, C, s, d, h)$ is said to be *feasible* if all packets generated in this channel can be delivered within the delay constraint d. The feasibility of a real-time channel depends on the values of h, $TTRT$ and their relations to T, C, and d. Given $TTRT$, T, C, and d, [14] provides a scheme to calculate the bandwidth required for a real-time channel to be feasible.

3 FDDI-based Mobile Networks

We describe in this section an architecture of FDDI-based mobile networks and point out issues that must be solved before real-time communication service can be supported in these networks.

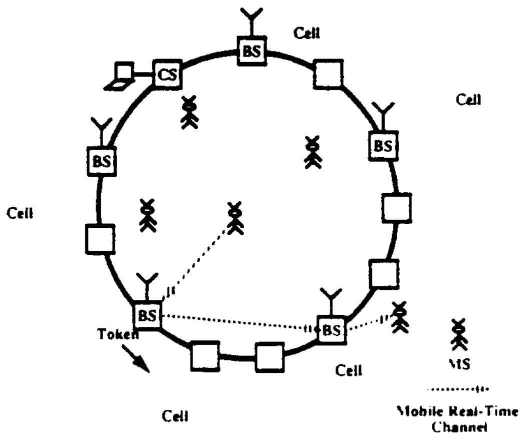


Figure 1: The architecture of FDDI-based mobile networks

3.1 Architecture for FDDI-based Mobile Networks

In this section, we propose an architecture of mobile systems based on FDDI networks. Its layout is depicted in Figure 1. Some components in this architecture are presented as follows. A mobile *control station* (CS) is a station that centrally manages the setups of mobile connections. A *mobile station* (MS) is a host that can move around while retaining its network connections without disruption. It is carried by a mobile user and serves as the user's interface to a mobile network. A *base station* (BS) is a wireless interface that connects to a wired network, in this case an FDDI network, as well as to the MSs within its wireless transmission range. Each BS is associated with a *cell*, which is the geographical area covered by its wireless transmission. Two cells are *overlapped* if they have a common area. Within this common area, an MS can communicate with two BSs. At any time, each MS has only one *local* BS, which is its primary data exchange interface. However, it may be able to exchange data with other BSs for a short interval.

When an MS moves from the cell of one BS into the cell of another, it will changes its local BS. This phenomenon is known as *handoff*. During a handoff, the path of data flow will change. A handoff time, t_h, is associated with a handoff. All packets generated before t_h are transmitted from the old local BS and all those generated after t_h are transmitted from the new local BS. According to [8], there are three types of handoffs: *hard*, *seamless*, and *soft* handoffs. For a seamless or soft handoff, the MS can simultaneously connect with two base stations for a while. This time period is called *degradation interval* and can last as long as several seconds [13]. Since soft handoffs are employed in the U.S. CDMA standard (IS-92) and considered in the European UMTS standard, we will assume soft handoffs in this paper.

3.2 Mobile Real-Time Communications

We propose to support real-time communications in an FDDI-based mobile network via *mobile real-time chan-*

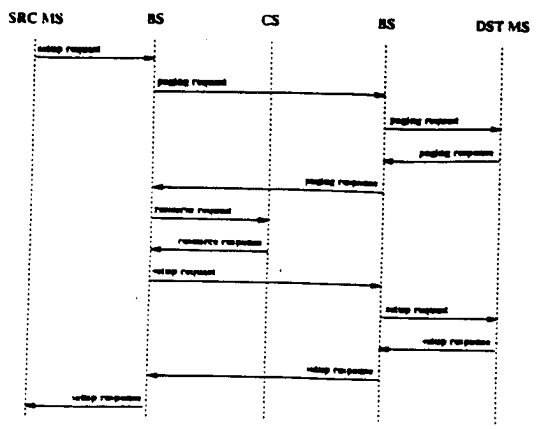


Figure 2: Real-time channel setup

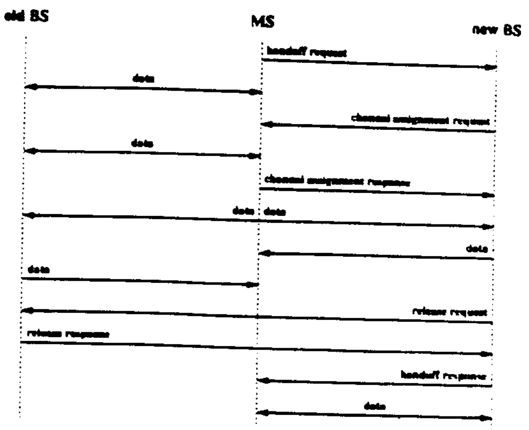


Figure 3: Real-time channel handoff

nels. A mobile real-time channel may have MSs as its source and destination. The data in a mobile real-time channel are first sent from the source MS to its local BS through a wireless channel. Then the BS transmits the data through the FDDI network to the local BS of the destination MS. Finally, this local BS delivers the data to the destination MS. By extending Definition 1, we give the following definition of mobile real-time channel.

Definition 2 A *mobile real-time channel* in an FDDI-based mobile network is characterized by a 5-tuple (T, C, s, d, h), which is defined as in Definition 1 except that 1) s is a mobile station, 2) the generation time or finish time of a packet is the time when its last bit reaches or leaves the local BS of s, respectively, and 3) h is the bandwidth allocated to the local BS of s.

For a mobile real-time channel, there are two importance stages of control: *setup* and *handoff*. We first show the setup of a real-time channel with MSs as both its source and destination. (See Figure 2.) When a mobile station wants to set up a channel, it sends to its local BS a *setup request* containing the destination MS's address and the channel's parameters T, C, d. The local BS broadcasts a *paging request*, trying to determine if the destination MS is in existence or if its power is on. The *paging request* is received by each BS, which broadcasts via the wireless medium to ask if the destination MS is in its cell. If the MS's power is on, it replies with a *paging response*. The response is broadcasted by the local BS to all other base stations via the FDDI backbone. Upon receiving the *paging response*, the local BS of the source MS sends a *resource request* to the CS with the channel's parameters. The CS decides whether to accept or reject the request depending on availability of resources. After the decision, the CS sends back (by broadcast) a *resource response*, with a channel ID if the request is accepted. If the *resource response* is positive, the BS of the source MS sends the *setup request* together with the channel ID to the destination MS, of course by way of a local BS. The destination MS replies with a *setup response*, which is delivered all the way to the source MS and the channel is ready. We do not exclude the possibility of the source or destination MS, or both, being moving to another cell during the setup process. With all messages between base stations being sent by broadcast, the above setup procedure is invulnerable of handoffs. Of course we need to assume that information regarding the requested channel will be passed from the old BS to the new BS during a handoff.

A handoff in these mobile networks is distributedly controlled, involving only the old local BS and the new local BS. It is also flexible, permitting the involved MS to exchange data with two BSs for a short period. The handoff process is depicted in Figure 3 and described as follows. When an MS detects that the neighboring BS's signal is stronger, it sends a *handoff request* to this BS. If this new BS can allocate a wireless channel within a certain interval, it sends a *channel assignment request* to the MS. Upon receiving the request, the MS sends back a *channel assignment response*. Afterward, the MS will exchange data with two BSs. Once the *channel assignment response* is received by the new BS, it sends a *release request* to the old BS; upon receiving a *release response* from the old BS, it sends a *handoff response* to the MS. Now the handoff is over. The MS has changed its local BS and will communicate with other stations through the new local BS.

3.3 Problems in Supporting Mobile Real-Time Channels

In the previous section, we showed the interactions between the MS and BSs during a handoff. However, an important question is left unanswered: how the data of the real-time channel will be handled in the old and new BSs so that the channel's QOS requirements can be satisfied. In this section, we first present a straightforward approach to the data handling during a source handoff of a real-time channel (T, C, s, d, h), then we identify some problems with this approach.

First consider the relationship of the handoff time and the token arrivals to the old and new BSs during a source handoff as depicted in Figure 4. Suppose a *handoff request* is received by the new BS and a wireless channel is properly assigned so that a *release request* is sent by the new BS at the token arrival $\bar{t}_{u-M+1}$. At the token arrival t_{u-M+1}, the old BS will send a *release response*

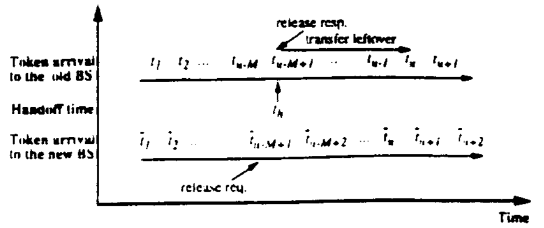

release resp.
transfer leftover
Token arrival to the old BS
t_1 t_2 … t_{u-M} t_{u-M+1} … t_{u-1} t_u t_{u+1}
Handoff time
t_h
Token arrival to the new BS
$\bar{t}_1$ $\bar{t}_2$ … $\bar{t}_{u-M+1}$ $\bar{t}_{u-M+2}$ … $\bar{t}_u$ $\bar{t}_{u+1}$ $\bar{t}_{u+2}$
release req.
Time

Figure 4: Relationship between time instances

to the new BS. The old BS will not transmit any data generated after t_{u-M+1}, which is regarded as the handoff time, t_h. However, some previously generated real-time packets, called the *leftover*, still have to be transmitted from the old BS. Let the number of token arrivals needed to transmit the leftover be M ($M \geq 1$) so that t_u is the last token visit for the old BS to transmit the data of the channel in question.

A straightforward approach to the handoff handling is that when a handoff happens, the new BS

1) Sends a request RAF of bandwidth h to the bandwidth management process at $\bar{t}_{u-M+1}$.

2) Starts data transmission from $\bar{t}_{u-M+2}$.

And the old BS

1) Stops receiving packets generated after t_{u-M+1}.

2) Continues to transmit the leftover until t_u.

3) Sends a release RAF of bandwidth h to the bandwidth management process at t_{u+1}.

However, simple as this approach is, it has some problems. In the rest of this section, we present these problems.

3.3.1 Overhead Bandwidth Problem

In the above approach, we can see that there is an overlapped time period when both the old and new BSs are allocated bandwidth h. For this time period, the real-time channel occupies bandwidth $2h$. The extra bandwidth of h is called *overhead bandwidth*, which is formally defined as follows:

Definition 3 For a real-time channel (T, C, s, d, h), if two base stations are respectively allocated an amount μ and μ' of bandwidth in a same round of token rotation, then the ***overhead bandwidth for that round of token rotation*** is $\mu + \mu' - h$. The ***overhead bandwidth for the handoff*** is defined as the maximum overhead bandwidth among all individual token rotations during the handoff period.

For the straightforward approach, the overlapped period is quite long, *i.e.* M token rotations. The amount is very large as well, *i.e.* h. The problems are how to make the period shorter and the amount smaller.

3.3.2 Bandwidth Management Problems

If the FDDI bandwidth management scheme described in Section 2 is employed, the straightforward approach has some problems:

1) *Heavy burden.* For each handoff, there is one bandwidth allocation and one bandwidth release: four control frames are exchanged. If the handoff rate is high, the burden on the bandwidth management process is very heavy. The problem is to find a bandwidth management scheme so that h can be reallocated from the old BS to the new BS without the BMP's interference.

2) *Slow response.* According to the FDDI's bandwidth management scheme, in the best case, the requested bandwidth can only be granted one token rotation after a request is sent. Consider the time sequence in Figure 4. The value of M is unknown to the new BS at $\bar{t}_{u-M+1}$ because a *release response* can only be sent from the old BS at t_{u-M+1}. If the new BS sends a bandwidth allocation request to the BMP at $\bar{t}_{u-M+1}$, the overlapped time period will be long for a large M. Otherwise, the new BS cannot transmit data at $\bar{t}_{u-M+2}$, which is $\bar{t}_{u+1}$ when $M = 1$; in this case, the channel's delay constraint may be violated. The problem is to find a bandwidth management scheme so that the requested bandwidth can be granted within one token rotation.

3) *Overhead bandwidth under-utilization.* One possible solution to the heavy burden and slow response problems is to pre-allocate a bandwidth h to each BS. This approach is obviously inefficient. Suppose there are 10 BSs in a mobile network and suppose statistically at most three handoffs may occur simultaneously in the whole network. Then, instead of pre-allocating $10h$ bandwidth, $3h$ will be sufficient. The problem is to find a bandwidth management scheme that allows global sharing of the pre-allocated bandwidth.

3.3.3 Ordering Problem

According to Figure 4, the real-time data transmitted from the new BS will be generated after $t_h = t_{u-M+1}$. If the new BS starts transmission at the next token arrival, $\bar{t}_{u-M+2}$, when $M > 1$, the old BS has not stopped transmitting data at t_{u-M+2}. Since the data transmitted by the old BS is generated earlier than those transmitted by the new BS, the data in the real-time channel will be disordered. If the new BS only starts transmission after the old BS has stopped, we face another problem as described in the following section.

3.3.4 Delay Problem

If the new BS holds the real-time data until the old BS has finished the leftover, then some data generated at the new BS may violate the delay constraint, as explained below. Suppose the number of token visits needed to transmit the leftover, M, is greater than 1. The total amount of leftover L satisfies $(M-1)h < L \leq Mh$. Suppose $L < Mh$. Suppose some data are sent by the source MS to the new BS before t_u. Consider the first part of these data which are no more than $Mh - L$. These data immediately follow those leftover at the old BS. Should no handoff have taken place, they would have been sent to the old BS and be transmitted from the old BS at t_u. But because of the handoff, these data are sent to the new BS instead, and will be held from transmission until $\bar{t}_{u+1}$. If the deadline for transmitting the mentioned data is t_u, the delay constraint is violated.

```
Overhead bandwidth management for each BS
IN CASE OF
• Bandwidth pool (BPF) control frame is received
    guard := the value in the BPF frame:
    IF overhead bandwidth $h_o$ requested by MAC
        IF guard $\geq h_o$
            guard := guard $- h_o$;
        ELSE reject the request
    ELSE IF overhead bandwidth $h_o$ released by MAC
        guard := guard $+ h_o$;
    Send a BPF control frame with the new value of
        guard to the downstream BS:
```

Figure 5: Overhead bandwidth management

3.3.5 Destination Problem

When a handoff occurs to the destination MS of a real-time channel, there is a *rerouting* problem. Before the handoff, the destination MS receives data from the old BS: but afterward, it receives data from a new BS. In an FDDI-based mobile network, the source station of the real-time channel should be able to notice this change and route the data to the new BS of the destination MS. The problem is how to make the rerouting.

4 Dynamic Bandwidth Management Scheme

In this section, we propose a *dynamic* bandwidth management scheme to solve the problems of Section 3.3.2. This scheme will be implemented in the SMT modules of BSs and the CS. It has following strengths: 1) bandwidth h can be reallocated directly from the old BS to the new BS without burdening the BMP: 2) the demands of overhead bandwidth can be satisfied immediately and some pre-allocated overhead bandwidth can be shared network-wide; 3) it is fully compatible with the FDDI bandwidth management scheme. Only BSs and the CS participate in this new bandwidth management scheme: all other FDDI stations and the BMP still use the original scheme.

The basic idea of this scheme is to make the CS a *proxy* of bandwidth management. Instead of requesting bandwidth by each BS individually, the CS requests bandwidth for all BSs. The bandwidth managed by the CS consists of two parts: the *normal* bandwidth used by the BSs off handoff periods and the *guard* bandwidth used as overhead bandwidth during handoffs. We show how to manage these two types of bandwidth as follows.

In order to share a pre-allocated bandwidth, a global variable *guard* is maintained distributedly by the BSs: it indicates the amount of *guard* bandwidth still available for use as overhead bandwidth. The most updated value of this variable is known by SMT modules of the BSs before each token arrival. When a BS needs some overhead bandwidth at a token arrival, it can test whether the needed amount is available. To implement this scheme, at each token arrival, a BS transmits a control frame, *bandwidth pool* (BPF), to the next BS downstream. This frame is very small, containing only the value of *guard*

```
Normal bandwidth reallocation in a BS
IN CASE OF
• Wanting to reallocate bandwidth to another BS:
    Send a BRF control frame to that BS;
    Decrease the bandwidth allocated to this station:
• Receiving a BRF control frame:
    Increase the bandwidth allocated to this station:
    Notify the MAC module of the reallocation:
Normal bandwidth reallocation in the CS
IN CASE OF
• Resource request received during a channel setup:
    Calculate the bandwidth needed for this channel:
    Send a RAF request to BMP for the bandwidth:
• RAF response received from the BMP:
    IF the requested amount is granted
        Send a BRF control frame to the requesting BS:
```

Figure 6: Bandwidth reallocation management

variable. When this frame is received by a BS, this value is examined and modified if some overhead bandwidth is needed. Then the new value is sent to the next BS in the downstream via another BPF frame. The detailed operations of this scheme are depicted in Figure 5.

When a mobile real-time channel is set up, an amount of normal bandwidth is allocated to the CS by the BMP. The bandwidth is then reallocated to the requesting BS by the CS. During a handoff, this normal bandwidth is reallocated directly from the old BS to the new BS. This reallocation scheme alleviates most of the BMP's burden in managing BSs' bandwidth. To implement this scheme, a control frame, *bandwidth reallocate* (BRF), is transmitted from the station releasing the bandwidth to the station accepting the bandwidth. It has one data field, specifying the amount of the reallocated bandwidth. The bandwidth is reallocated once the accepting station receives this frame. The details of the operations are depicted in Figure 6.

5 Handoff Control for Mobile Source

In this section, we first present some solutions to the source handoff problems mentioned in Section 3.3. Then we propose a handoff protocol for implementation within the MAC layer of a BS. Finally, we revisit the QOS problems after some assumptions are dropped.

5.1 QOS Guarantee in Source Handoff

In Section 3.3, there are two problems concerning the QOS during a source handoff: ordering problem and delay problem. For the ordering problem, an easy solution is to make the destination capable of packets reordering. Real-time packets transmitted in FDDI frames should bear some sequence numbers and some buffers should be provided in the destination. If packets are received out of sequence, they are put into buffers until the preceding packets arrive. We first assume the destination has this reordering capability. Then in Section 5.3, we show the ordering problem can still be solved even if this assumption is dropped.

It is easy to see that the straightforward approach can guarantee the delay constraint. The packets transmitted by the new BS will be generated after $t_h = t_{u-M+1}$ and the new BS will be allocated bandwidth h at $\bar{t}_{u-M+2}$, the first token visit after the packets are generated. Since the original bandwidth h is so allocated that the real-time channel is feasible, we know the data transmitted from the new BS will not be delayed. The data transmitted from the old BS will not be delayed as well because the handoff does not affect their transmissions. However, the straightforward approach is not efficient because it may allocate an overhead bandwidth long before it is needed. The problem we will solve in the rest of this section is to find out the last token arrival for the new BS to start data transmission without delay. First we define the *latest cleanup time* in a real-time channel.

Definition 4 For a mobile real-time channel (T, C, s, d, h), let $Q = \langle t_g, t_0, t_1, \ldots \rangle$, where t_g is the starting time of packet generation and $t_0, t_1, \ldots$ are the token visits to the local BS of s after t_g. A time instant $t_c \in Q$ is said to be a *clean-up time* for the channel if after the token visit of t_c, the local BS of s has delivered all packets generated before t_c. The *latest clean-up time* for time t, denoted by $LCT(t)$, is the latest among all the clean-up times that are not later than t. For convenience, regard t_g as a clean-up time so that $LCT(t)$ is meaningful for all $t \geq t_g$.

According to this definition, we can see that all token visits between $LCT(t)$ and t (not include them) will use up the allocated bandwidth h. Using this definition, we derive the following theorem to specify the last token arrival for the new BS to start transmission.

Theorem 2 *If the handoff time $t_h = t_{u-M+1}$ and the new BS is allocated h for its first data transmission, the latest token arrival for the new BS to start transmission without delay is $\bar{t}_{u+1}$ if $M = 1$ or $\bar{t}_u$ if $M > 1$.*

Proof. Let t_g be the packet generation starting time and t'_i $(i \geq 0)$ be the token arrivals to transmit these packets. Bandwidth h allocated by any scheme, such as in [1, 14], ensures that all packets will be transmitted without delay if

$$t'_i \leq t_g + (i+2)TTRT - h \tag{1}$$

If $M = 1$, $t_h = t_u$. According to Theorem 1, we have $\bar{t}_{u+i+1} < t_{u+i+1} \leq t_u + (i+2)TTRT - h$ for any $i \geq 0$. By Eq.(1), we know data generated after t_h will not be delayed if transmitted at $\bar{t}_{u+1}$. Otherwise, if $M > 1$, let $LCT(t_h)$ be the latest cleanup time for t_h and let t_g be the generation time of the first packet transmitted by the new BS. Suppose there are M' token visits to the old BS after $LCT(t_h)$ (not include it) and before t_u (include it). It is easy to see $M' \geq 2$. According to Definition 4, all token visits between $LCT(t_h)$ and t_u (not include them) use up bandwidth h and all data transmitted by these token visits are generated after $LCT(t_h)$. According to [14], to satisfy the throughput requirement, h should be no less than $(TTRT/T)C$. Since the packets transmitted from the new BS are generated after those transmitted

```
LOOP BEGIN
IN CASE OF
• Handoff request received from an MS
   Call the move-in module;
   Thereafter transmit real-time packets using
      bandwidth h at each token arrival;
• Release request received from another BS
   Call the move-out module;
LOOP END
```

Figure 7: The main program of the handoff protocol

from the old BS, we have

$$\begin{aligned} t_g &\geq LCT(t_h) + (M'-1)h(T/C) \\ &\geq LCT(t_h) + (M'-1)TTRT \end{aligned} \tag{2}$$

According to Theorem 1, we have

$$\bar{t}_{u+i} < t_{u+i} \leq LCT(t_h) + (M'+i+1)TTRT - h \tag{3}$$

for any $i \geq 0$. From Eq.(2) and Eq.(3), it is easy to derive that $\bar{t}_{u+i} \leq t_g + (i+2)TTRT - h$. By Eq.(1), we know if the bandwidth h is allocated at $\bar{t}_u$, all packets generated at or after t_g will be transmitted without delay. ■

5.2 Handoff Protocol for Mobile Source

Based on Theorem 2 of the previous section and the bandwidth management scheme of Section 4, we propose the following new approach to data handling in a source handoff. When a handoff occurs, the new BS do the following:

1) Receive a ***release response*** from the old BS and decide the value of M.

2) Request an overhead bandwidth $h_o = h$ out of the guard bandwidth and start transmission at $\bar{t}_{u+1}$ if $M = 1$ or at $\bar{t}_u$ if $M > 1$.

3) Release overhead bandwidth h_o and acquire the reallocated bandwidth h from the old BS at $\bar{t}_{u+2}$.

4) Transmit data using bandwidth h at and after $\bar{t}_{u+2}$.

The old BS do the following:

1) Stop receiving packets generated after t_{u-M+1}.
2) Continue to transmit the leftover until t_u.
3) Release bandwidth h for reallocation at t_{u+1}.

The main program of the handoff protocol to implement this approach is depicted in Figure 7. If no handoff happens, when the token arrives, the handoff protocol transmits data using bandwidth h as usual; when a source handoff takes place, depending on whether the source enters the cell of the BS (a ***handoff request*** received from the MS) or leaves the cell of the BS (a ***release request*** received from another BS), the main program calls either the ***move-in*** or the ***move-out*** module. The old BS's ***move-out*** module cooperates with the new BS's ***move-in*** module to ensure a smooth handoff.

The interactions between the ***move-out*** and ***move-in*** modules are explained as follows. Immediately after it is evoked, the ***move-in*** module allocates a wireless channel and informs the MS of the new channel. Afterward, at the next token arrival, it sends out a *release request* to

```
LOOP BEGIN
• When token arrives
    IF this is the first token arrival after the handoff
      Stop receiving data from the source MS;
      $M := \max(\lceil(\text{the amount of leftover})/h\rceil, 1)$;
      $OH := Mh$ - the amount of leftover;
      Transmit a release response with the values of $M$,
          $OH$ and sequence number of the last received
          packet to the new BS;
    Transmit the leftover to the destination of the
        real-time channel using bandwidth $h$;
    Release the token;
    IF the leftover have been depleted
      Notify the SMT module to reallocate normal
          bandwidth to the new BS;
      Return to the main program;
LOOP END
```

Figure 8: The *move-out* module of the handoff protocol

the *move-out* module of the old BS. At the first token visit after the *release request* is received, the *move-out* module in the old BS stops receiving data from the MS and sends out a *release response* to the *move-in* module. The parameters in the *release response* specify when the new BS should start transmission and from which packet the new BS should transmit. Additional information such as channel ID and destination address is sent to the new BS from the MS via a *handoff request*. The *move-out* module continues to transmit the leftover at bandwidth h until they are depleted. Then before the next token arrival, it notifies the local SMT module to release normal bandwidth h. Based on the parameters of the *release response*, the *move-in* module in the new BS decides how many token visits should be passed before it starts transmission. Before the first transmission that will incur overhead bandwidth, the *move-in* module requests the local SMT module to allocate some overhead bandwidth out of the guard bandwidth. After bandwidth h is reallocated from the old BS, the *move-in* module asks the local SMT module to release the overhead bandwidth. Now the handoff is over and the control is returned to the main program.

5.3 Further Discussions

In this section, we will discuss what changes should be made to the handoff protocol when two assumptions, destination reordering and soft handoff, are dropped.

If the destination cannot reorder packets, the new BS cannot send packets directly to the destination until the old BS has depleted the leftover. However, the delay constraint may be violated as shown in Section 3.3.4. To overcome this dilemma, the handoff protocol should do *packet relay*. Before the leftover are depleted in the old BS, the new BS starts to relay some packets to the old BS, which will be sent later to the destination by the old BS. If $M = 1$, no relay is needed because the new BS starts after t_u. Otherwise, if $M > 1$, it is enough to relay data of OH units, the amount of data to fill up the last transmission of the old BS. By relaying these data, all

```
Allocate a wireless channel;
Send a channel assignment request to the MS;
Wait until channel assignment response is received;
LOOP BEGIN
IN CASE OF
• Token arrival
    IN CASE OF
      • this is the first arrival after
            channel assignment response
        Send a release request to the old BS;
      • $M > 2$ and this is one of the $M - 2$ arrivals
            after release response
        Do nothing;
      • $M > 1$ and this is the $(M-1)$th or $M$th arrival
            after release response
        Transmit data using $h_o$ if available;
      • $M = 1$ and this is the first arrival after
            release response
        Transmit data using $h_o$ if available;
    Release the token;
    IF $M > 2$ and this is the $(M-2)$th arrival after
        release response
      Ask SMT module for overhead bandwidth $h_o$;
• Release response received from the old BS
    $M, OH$ := the values in release response;
    Discard the packets up to the sequence number in
        release response;
    IF $M = 1$ or $M = 2$
      /* transmission starts at next token visit */
      Ask SMT module for overhead bandwidth $h_o$;
• Notified by SMT module of bandwidth reallocation
    Ask SMT module to release overhead bandwidth $h_o$;
    Return to the main program;
LOOP END
```

Figure 9: The *move-in* module of the handoff protocol

the data transmitted by the new BS to the destination at $\bar{t}_{u+1}$ would be transmitted from the old BS at t_{u+1} should no handoff have taken place. Since $\bar{t}_{u+1} < t_{u+1}$, no packet transmitted by the new BS to the destination is late.

If the assumption of soft handoff is dropped, the MS cannot maintain data exchange with both the old and the new BS. Therefore, we cannot control the data flow and make $t_h = t_{u-M+1}$. However, t_h should satisfy $\bar{t}_{u-M} < t_h \leq \bar{t}_{u-M+1}$. Similar to Theorem 2, it can be proved that the new BS should start transmission at $\bar{t}_u$ no matter $M = 1$ or $M > 1$.

6 Handoff Control for Mobile Destination

In this section, we give solutions to the problem caused by mobile destinations. Two approaches, multicasting and rerouting, are addressed and compared.

If the destination of a real-time channel is an MS, we have to ensure that data still be received by the destination even if it has changed its local BS. Specifically, there are three requirements: 1) No packet is duplicatedly delivered from both the old and the new BS of the destination MS; 2) No packet is lost; 3) All packets are delivered to the destination MS once the BSs receive them, i.e. no delay in delivery. To meet these requirements, we may adopt one of two approaches: *multicasting* or *rerouting*. For the first approach, the source station multicasts the packets to all BSs. Upon receiving the packets, the BSs decide which packet should be delivered to their MSs. For the second approach, packets are only sent to the local BS of the destination MS. The BSs deliver all received packets to their MSs. The first approach does not incur extra traffic but it imposes more load on the BSs because they have to receive and process the packets not intended for them. Its main advantage is no need for rerouting. The implementations of the two approaches are presented as follows.

In the multicast approach, after a *release request* is received by the old BS of the destination MS, in the next token visit, it sends out a *release response* to the new BS and then stops delivering any packet received afterward. Upon receiving a *release response*, the new BS starts to deliver the packets received after this response. There is no packet loss or duplication because the packets transmitted from the source before the *release response* are delivered from the old BS and those transmitted afterward are delivered from the new BS. No packet is delayed in delivery to the destination MS as well because the new BS does not hold any packet received after the *release response*. In the rerouting approach, after a *release request* is received, in the next token visit, besides sending a *release response* to the new BS, the old BS also broadcasts a *destination reroute* control frame (DRF) with this channel's ID to all stations. Upon receiving this control frame, the source station starts data transmission to the new BS. No packet is duplicated, lost or delayed because any packet is transmitted to only one BS and delivered promptly. Since the DRF frames consume some bandwidth, when handoff rate is high, this approach is discouraged.

7 Conclusion

We have addressed two issues: (1) how to build a mobile network using the FDDI as a backbone and (2) how to provide real-time communication service on such a network. To the best of our knowledge, this is the first paper on these topics. In particular, we described an architecture of FDDI-based mobile networks, identified some problems in supporting real-time communications, and proposed possible solutions. Our solutions are compatible with the current FDDI standards in the sense that only base stations and the control station will need our schemes. Those stations having nothing to do with mobility will need no modification. In particular, the bandwidth management process remains unchanged. A complete version of this paper is available as a technical report [12], which includes an algorithm to minimize the overhead bandwidth.

References

[1] G. Agrawal, B. Chen, and W. Zhao, "Local synchronous capacity allocation schemes for guaranteeing message deadlines with the timed token protocol," in *Proc. INFOCOM*, pp. 186–193, March 1993.

[2] ANSI X3T9, *FDDI Media Access Control-2 (MAC-2) - draft proposed*. American National Standard, ANSI X3T9/92-120, 1992.

[3] ANSI X3T9, *FDDI Station Management (SMT) - draft proposed*. American National Standard, ANSI X3T9/92-067, 1992.

[4] T. Balch and R. C. Arkin, "Communications in reactive multiagent robotic systems," *Autonomous Robots*, vol. 1, pp. 27–52, 1994.

[5] B. Chen and W. Zhao, "Properties of the timed token protocol," Technical Report TAMU 92-038, Department of Computer Science, Texas A&M University, 1992.

[6] D. Ferrari and D. C. Verma, "A scheme for real-time channel establishment in wide-area networks," *IEEE Journal on Selected Areas in Communications*, vol. SAC-8, pp. 368–379, April 1990.

[7] D. J. Goodman, "Cellular packet communications," *IEEE Transactions on Communications*, vol. 38, no. 8, pp. 1272–1280, August 1990.

[8] B. Jabbari, G. Colombo, A. Nakajima, and J. Kulkarni, "Network issues for wireless communications," *IEEE Communications Magazine*, pp. 88–98, January 1995.

[9] R. Jain, *FDDI Handbook: High Speed Networking Using Fiber and Other Media*. Addison Wesely, 1994.

[10] M. J. Johnson, "Proof that timing requirements of the FDDI token ring protocol are satisfied," *IEEE Transactions on Communications*, vol. Com-35, no. 6, pp. 620–625, June 1987.

[11] D. D. Kandlur, K. G. Shin, and D. Ferrari, "Real-time communications in multi-hop networks," in *Proc. IEEE International Conference on Distributed Computing Systems*, pp. 300–307, 1991.

[12] T.-H. Lai, Y. Yang, and M. T. Liu, "Real-time communication in FDDI-based mobile networks," *Technical Report (postscript file available from the authors)*, 1995.

[13] S. Tekinay and B. Jabbari, "A measurement-based priorization scheme for handovers in mobile cellular networks," *IEEE Journal on Selected Areas in Communications*, vol. 10, no. 8, pp. 1343–1350, Oct. 1992.

[14] Q. Zheng and K. G. Shin, "Synchronous bandwidth allocation in FDDI networks," *to appear in IEEE Transactions on Parallel and Distributed Systems*, 1995.

E. M. T. Liu,
"Interconnection and Protocol Converter,"
accepted; to appear in *Advances in Computers*,
vol.40, Academic Press, January 1996.

Network Interconnection and Protocol Conversion[1]

Ming T. Liu
Department of Computer and Information Science
The Ohio State University
2015 Neil Avenue
Columbus, OH 43210-1277

[1]Research reported herein was supported by U.S. Army Research Office, under contract Nos. DAAL03-91-G-0093 and DAA03-92-G-0184. The views, opinions, and/or findings contained in this paper are those of the authors and should not be construed as an official Department of the Army position, policy or decision.

Contents

1 Introduction

Decades ago, the first step in distributed data processing was to use telephone lines and terminals to access computers from remote locations. Later, the development of higher-speed and reliable packet-switching networks made it possible to interconnect a large number of computers and terminals. Since then, the data communication world has undergone a major change, from an ad hoc array of systems to a more planned and integrated set of facilities designed both for data communications and for public use. In fact, as early as 1970s, data networks have been built to meet many user needs. For single organizations, these data networks are often private ones, built with a technology optimized for the specific application. At that time, many private networks have been set up using various technologies; to name a few of them: SITA (Hirsch, 1974), SWIFT (Lapidus, 1976), ARPANET (Roberts and Wessler, 1973; Karp, 1973), CYCLADES (Pouzin, 1973), ETHERNET (Metcalfe and Boggs, 1976), PRNET (Kahn, 1975), and SPYDER (Fraser, 1974). For communication between organizations, networks are being built by licensed carriers. For example, since early 1970s, there have been many such licensed carriers as TELENET (Roberts, 1975), DATAPAC (Clipsham et al., 1976), TYMNET (Rinde, 1976). In the rest of the world, the Post, Telegraph and Telephone Authority (PTT) in each country has a near monopoly on such services; special public data networks are built in many countries, including TRANSPAC (Danet et al., 1976) in France, EURONET (Davies, 1976) for inter-European traffic, DDX (Nakamura et al., 1976) in Japan, EDS (Helsel and Spadafora, 1976) in Germany and NPDN (Larsson, 1976) in Scandinavia. Most of them use packet-switching technology.

It is a common user requirement that a single terminal and access port should be able to access any computing resource the user may desire - even if the resource is on another data network. From this requirement, there is a clear need to have data networks connected together. By the same token, the providers of data network services would like to have their networks used as intensively as possible; thus they also have a strong motivation to connect their data networks to others. From the implementation viewpoint, there can be some considerable complications in connecting networks of widely different technologies such as circuit-switched and packet-switched networks. In addition to technical issues, network interconnection raises many legal and political issues. While the technical issues generally revolve around mechanisms for achieving interconnection and their performance, the legal and political issues are at least as complex as the technical ones. For example, questions like "*can private networks interconnect to each other or must they do so through the mediation of a public network?*", "*how is privacy to be protected?*", "*what kinds of charging and accounting policies should apply to multinetwork traffic?*" are just a few of them. It is impossible to answer all of these questions all at once. Therefore, the main focus of this article is on the technical side of network interconnection problems.

From a pure technical viewpoint, due to rapid advances of computer technology in recent years, the advent of large numbers of microcomputer workstations in the workplace has brought the *information age* close to reality. A network that allows easy access to remote resources and easy distribution of information enhances the productivity of users as well as their organizations. The need of achieving interoperability between equipments and systems of different makes has brought about conventions for exchanging data, known as communication protocols, or simply protocols.

Protocols are specific to closed user communities. Creating an environment that supports effective communications among diverse sets of such communities is no simple task. Because different networks use different architectures and protocols, it is impractical to consider merging them into a single network. Rather, what is needed is the ability to interconnect various networks so that any two stations on any of the constituent networks can communicate. Building networks of networks or interconnecting existing networks is a natural approach until overall standardization takes over.

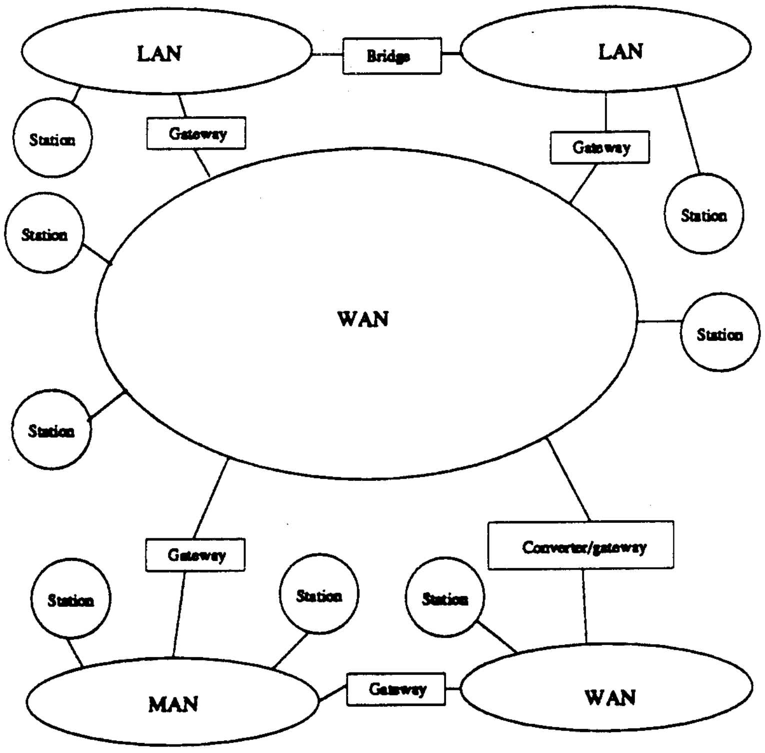


Figure 1: Network Interconnection

Nevertheless, existing systems will remain long after new standards are promulgated.

This article presents both an expository survey on network interconnection and the results of our approach to protocol conversion. The first part (Sections 2 to 5) considers various issues and approaches to network interconnection (including low to medium speed networks as well as high-speed networks). The second part (Section 6 to 9) deals with formal approaches to protocol conversion, the Synchronizing Transition Set (STS) algorithm, and protocol conversion in Extended Finite-State Machine (EFSM) models as well as in multimedia networks.

Throughout this article, the terms, *constituent network*, *subnetwork*, or simply *subnet*, are often used inter-changeably. They have the same meaning unless specifically distinguished otherwise.

1.1 Network Interconnection

Fig. 1 illustrates an abstraction of interconnecting multiple networks, where various autonomous networks are interconnected by a backbone network through *gateways*. In ISO terminology, a gateway that interconnects two networks is known either as an *intermediate system* (IS) or an *interworking*

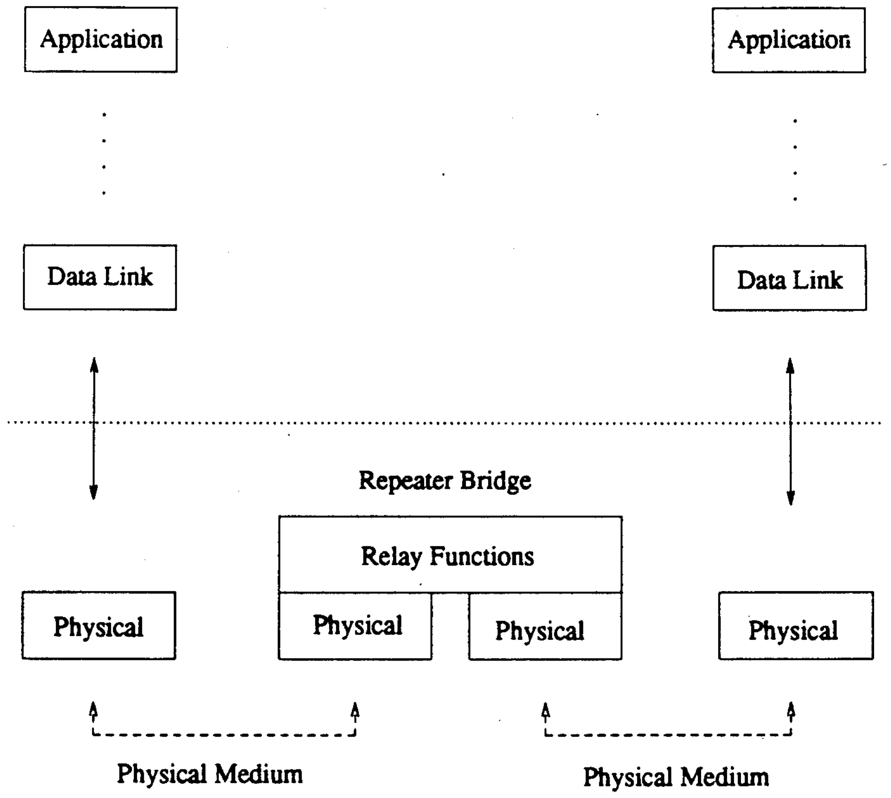


Figure 2: A Repeater Bridge

unit (IWU). Moreover, since one of the major functions performed by a gateway is routing, it is sometimes referred to as a *router*.

Each constituent network supports communication among a number of attached devices, using gateways to provide communication paths. The attached devices are labeled as *stations* to distinguish

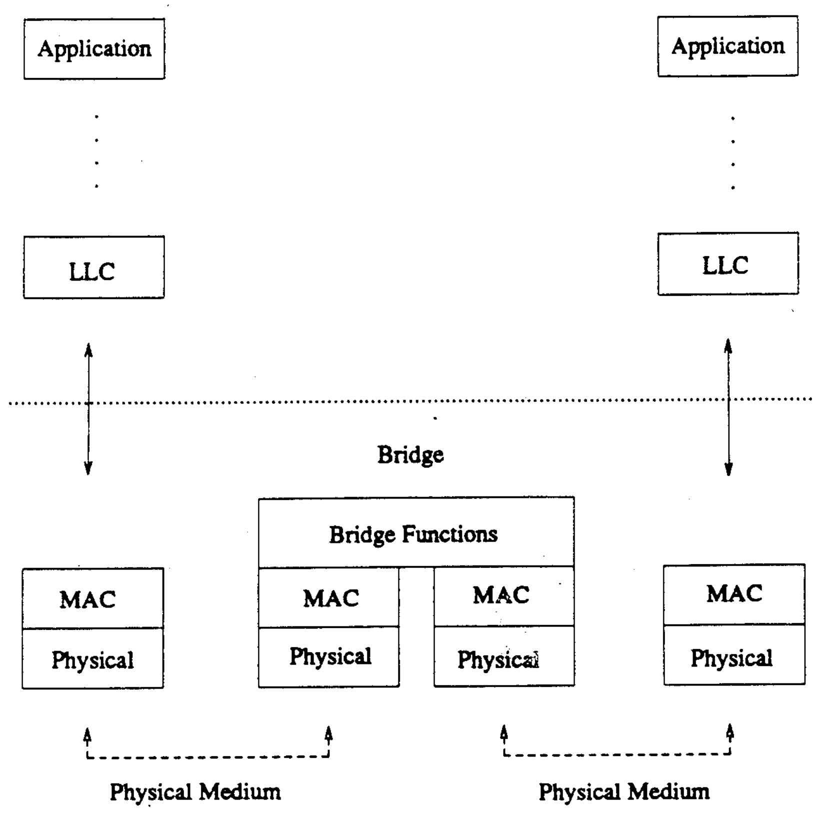


Figure 3: A MAC Relay Bridge

themselves from the internal *nodes* of a network. Again in ISO terminology, a station in Fig. 1 is called DTE (data terminal equipment) and an internal node is called DCE (data circuit-terminating equipment).

Network interconnection can be accomplished through a variety of strategies (Green, 1986; Lloyd and Kirstein, 1975; Cotton, 1978; Chou, 1985; Perlman, 1992): layer substitution, addition of extra layers, bridging individual layers, etc. Discussion on these connection strategies is presented in Section 3. On the other hand, in terms of interconnection functions provided, gateways can be classified either as bridges or as converters. In general, bridges are used in homogeneous networks as a packet relay point with minimum protocol conversion functions, while converters need to perform major protocol conversion functions in addition to relaying packets. Converters are used to interconnect heterogeneous networks.

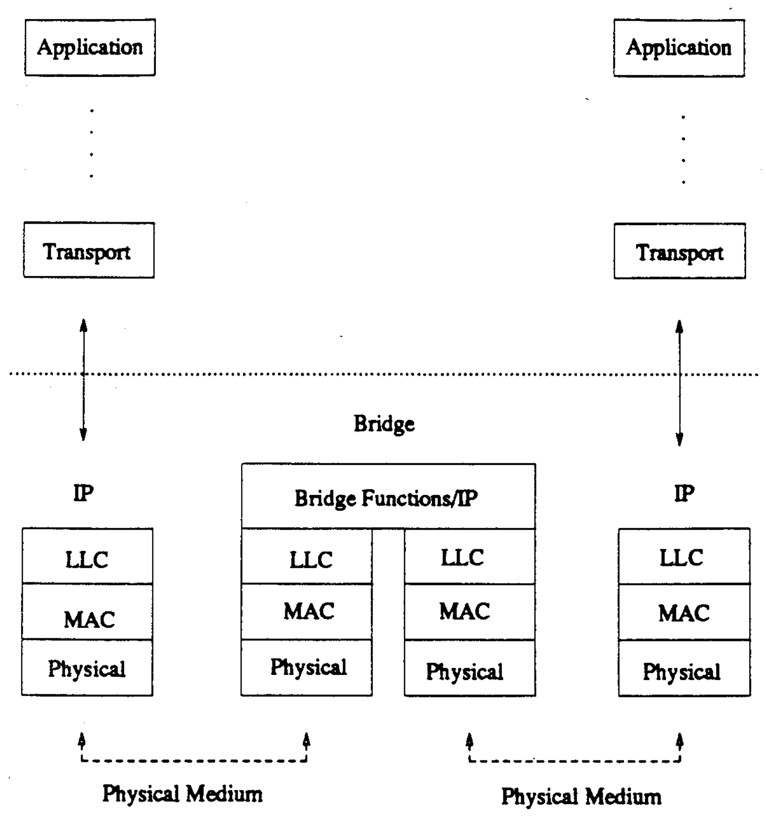


Figure 4: An Internet Bridge

1.1.1 Bridges

Bridges are used to interconnect homogeneous networks at the node (DCE) level, which implies, at minimum, that the networks have a common network access interface such as X.25. On the other hand, internetworking through bridges does not implies that the networks have the same internal protocols. A common network access interface, in the context of this section, means there is a standardized format for packets of the interconnected networks. In an optimal situation, a bridge is used merely as a packet relay between networks. With the exception of possible address space expansion, the stations in a network do not need to be aware that there are multiple networks. Thus, when it is designed properly, no changes need to be made in the station software. For the obvious reason of simplicity, using bridges to interconnect *similar* networks has lower cost and higher performance than using converters. In terms of the ISO model, bridges can be placed on top of either of the three lowest layers depending upon the degree of similarity among the interconnected networks. The lower the layer a bridge is placed on, the less complex a bridge becomes.

A bridge is a simplified gateway. A simplest example is the repeater on a baseband network (Fig. 2), which amplifies and retransmits all signals to extend the length of the baseband cables. However, this is not a true multiple network system. Within the context of the IEEE 802 standard, a bridge shown in Fig. 3 is referred to as a *MAC-level* relay in 802.4 and 802.5. In such a structure, a globally administrated set of MAC station addresses is used across multiple homogeneous LANs. In the DoD standard Internet, the bridge function is placed below the transport layer as shown in Fig. 4, where the bridge is involved in the routing functions among networks. A specialized internet protocol, DoD IP, is used to accomplish the interconnection. More discussion on DoD IP is given in Section 4.2. Other examples of using bridges can be found in (Thornton and Christenson, 1983; Marchall, 1983; Estrin and Carrico, 1982). In general, when a *bridge* is placed on a layer higher than the network layer, it becomes a protocol converter/adaptor due to the increased complexity involved in the bridge function.

In recent years, the bridging mechanism is most commonly used to connect multiple homogeneous local area networks. The main functions of a bridge between two homogeneous LANs can be summarized as follows (Stallings, 1994):

- Reads all frames/packets transmitted on one network and accepts those addressed to another.
- Uses the medium access control protocol of the destination network to retransmit the frames/packets.
- Makes no modification to the content of format of the frames it receives, nor does it encapsulate them with an additional header.
- To meet peak transmission demands, a bridge needs to buffer frames to be retransmitted. It performs a store-and-forward function.
- Needs to have addressing and routing capability when necessary.
- Increases the physical extent of interconnected networks.
- Increases the maximum number of connected stations in interconnected networks.

In addition, a well-designed bridge should have desirable characteristics as follows (Hawe et al., 1984):

- Minimum Traffic - Only traffic generated by users should exist on the interconnected networks. No traffic results from complex routing algorithms.
- No duplicates - A bridge should not cause duplicated frames to be delivered to the destination.
- Sequentiality - No permutation of the ordering as transmitted by the source station.
- High performance - Bridges should be able to process frames at the maximum rate at which they can be received.
- Frame lifetime limit - Frames should not be allowed to exist in the interconnected LANs for an unbounded time.
- Low error rate - Bridge should not increase the error rates of the original LANs.
- Others such as low congestion loss and general topology.

The best known technique for forwarding frames in bridged networks is based upon the use of hierarchical organized address space. In this scheme, the address space is partitioned into fields describing in which LAN in the hierarchy the station resides. However, there are many problems of this scheme as reported in (Hawe et al., 1984). To name a few of them: the topology is restricted to a rooted tree; an end node must be told its own address either manually or by dynamic binding which requires the existence of a complex protocol; the depth of the physical hierarchy is pre-defined by the number of fields in the address space, etc. To overcome these problems, various routing schemes and algorithms have been examined by many researchers (Pitt and Winkler, 1987; Hawe et al., 1984; Ahmad and Halsall, 1993; Sunshine, 1983; Baratz and Jaffe, 1986; Wong et al., 1987; Ryder, 1983; Shoch et al., 1980). More in depth discussion of internetwork routing is presented in Section 2.2.

1.1.2 Converters

A converter is needed when the subnetworks to be interconnected have nontrivial *protocol mismatches*. In (Green, 1986), different internetwork architectures were analyzed and situations in which conversion must be performed were discussed. Green classified protocol mismatches into *soft* and *hard* mismatches. Soft mismatches are those which result in reduced (and probably unacceptable) expectations in the performance of the two protocols being converted, such as occasional loss of messages, buffer overflow, lower availability, etc. Hard mismatches are those which render the system completely inoperable. Examples of hard mismatches include noncompletion of path and deadlock, etc. Protocol conversion is to resolve both types of protocol mismatches. Depending on the degree of mismatches, protocol conversion complexity can range from trivial to impossible.

In a typical network, data of a fixed maximum size are accepted from a source station, together with a formatted destination address which is used to route the data in a store and forward fashion. The transmit time for this data is usually dependent upon internal network parameters such as communication media data rates, buffering and signaling strategies, routing, propagation delays, etc. In addition, some mechanism is generally preset for error handling and determination of status of subnetworks. Some of the typical requirements of a converter can be as follows (Cerf and Kahn, 1974):

- Each subnetwork may have distinct ways of addressing the receiver, thus requiring that a uniform addressing scheme be created which can be understood by each individual subnetwork.
- Each subnetwork may accept data of different maximum sizes, thus requiring subnetworks to transmit in units of the smallest maximum size (which may be impractically small) or requiring procedures that allow data crossing a network boundary to be reformatted into smaller pieces.
- The success or failure of a transmission and its performance in each subnetwork is governed by different time delays in accepting, delivering and transporting the data. This requires careful development of internetwork timing procedures to insure that data can be successfully delivered through various subnetworks.
- Within each subnetwork, communication may be disrupted due to unrecoverable mutation

of data or missing data. End-to-end restoration procedures are desirable to allow complete recovery from these conditions.

- Status information, routing, fault detection, and isolation are typically different in each subnetwork. Thus, to obtain verification of certain conditions, such as an inaccessible or dead destination, various kinds of coordination must be invoked between the subnetworks.

In terms of network configuration, protocol conversion can be implemented either through layer substitution or the intermediate converter model. More in depth discussion is provided in Section 3.2.

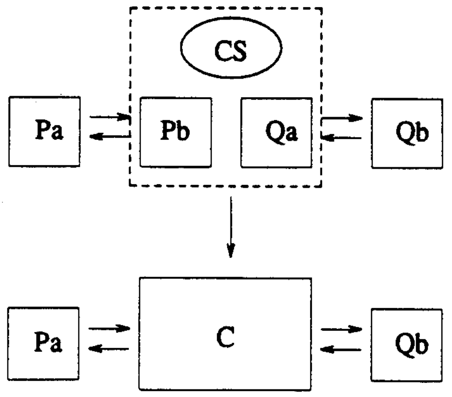
It would be extremely convenient if all the differences between subnetworks could be economically resolved by suitable interfacing at the network boundaries. For many of the differences, this objective can be achieved through a conversion function, as seen in many of the ad hoc converter algorithms (Francois and Potocki, 1983; Groenback, 1986; Rodriguez, 1988; Rose and Cass, 1987). However, both economic and technical considerations mandate simpler and more reliable interfaces, which cannot be achieved by ad hoc converters due to their complexity. To overcome this complexity problem, formal methods for protocol conversion have received more and more attention since mid-1980s. Discussion on the formal approaches is provided in the next sub-section; the detail is given in the second part of this article (Sections 6 to 9).

1.2 Protocol Conversion

To address the high complexity issue of ad hoc protocol conversion algorithms, many researchers started using formal approaches to resolve the protocol mismatch problems since 1980s. In general, formal methods use high level abstraction to formally define the protocol conversion problem. Thus, the complexity of constructing an abstract converter is subsequently reduced using various formal procedures. Formal protocol conversion has since become an important area of research in *protocol engineering* (Liu, 1989).

1.2.1 Conversion Specification Approach

Among those formal methods proposed by different researchers, there are in general two different approaches: *Conversion Specification* and *Service Specification*. To derive a protocol converter, the conversion specification approach takes directly as input the original protocol specifications and the given conversion specification, in a similar form, as illustrated in Fig. 5. Essentially, the conversion specification either tells how to translate the messages between protocols or specifies the ordering of the given protocol transitions to synchronize the execution of the given protocol entities. In other words, the conversion specification tells how to perform the protocol conversion directly for the given protocols. The algorithms proposed in (Okumura, 1986; Y. W. Yao and Liu, 1990a; Lam, 1986; Calvert and Lam, 1987; Lam, 1988; Shu and Liu, 1989; Shu and Liu, 1990; Shu and Liu, 1991; Chang and Liu, 1990b; Chang and Liu, 1990a) belong to this category.



P_a, P_b, Q_a and Q_b: protocol specifications
CS: conversion specification
C: final converter specification

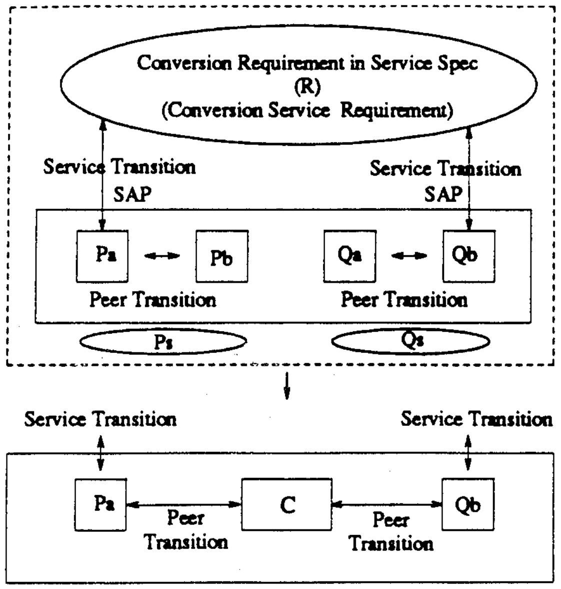
Figure 5: The Conversion Specification Approach

One of the shortcomings of the conversion specification approach is that the implementation details are involved at the early stage of design, thereby resulting in a more complex algorithm. In addition, there is no formal algorithm for generating the conversion specification, which is usually obtained in an ad hoc manner. This is a very serious drawback, especially for those complex protocols whose conversion specifications are hard to find in ad hoc ways. Examples of conversion specification approaches are discussed in Section 6.1.

1.2.2 Service Specification Approach

The service specification approach, which has received more and more attention in recent years, treats the implementation details as unknown *black boxes* at the early stage of design; it only depicts the final behavior of the overall protocol system as a to-be-found converter from the viewpoint of the protocol service users at the *SAP* (Service Access Point). The service specification is specified in terms of the ***Service Transitions*** to be executed at the SAPs. Fig. 6 shows the high level concept of the service specification approach. The algorithms proposed in (Calvert and Lam, 1989; Calvert and Lam, 1990a; Calvert and Lam, 1990b; Okumura, 1990; Yao and Liu, 1992) belong to this category.

Due to its high level of abstraction, finding a correct service specification at the protocol service user level is easier and more practical than finding the conversion specification in an ad hoc manner by examining the implementation of the given protocols. However, the difficulty of the service specification approach is, in general, that the service specification by itself does not tell directly how to perform the protocol conversion. As a result, the *information* on how to perform the conversion must be derived from the given service specifications and requirement. Since the conversion



P_a, P_b, Q_a and Q_b: protocol specifications
P_s, Q_s: service specifications
R : Conversion Service Requirement
C: final converter specification

Figure 6: The Service Specification Approach

information derived is not specific, the resulting converter may not be correct. Consequently, most of the algorithms in this category require a validation phase at the end to ensure the correctness of the protocol converter constructed. This not only complicates the construction procedure but also means that when the validation fails, the procedure has failed to find a converter for the given service specifications and requirement. Moreover, even when the validation phase may pass, the converter so constructed often covers only a subset of the properties and functionalities of the original protocols (Liu, 1990; Yao and Liu, 1992) even though the conversion service requirement does not specify the removal of any transitions. This *unspecified* lost of properties and functionalities, which often results in incorrect conversion, is due to the fact that the information on the existing protocol implementation is not taken into consideration during the converter construction process.

In (Jeng and Liu, 1992), efforts are made in an attempt to resolve the aforementioned drawbacks of the service specification approach. The details are presented in Sections 6.2 and 7.

2 Issues in Network Interconnection

Network interconnection faces many issues. As early as mid 1970s researchers have started to examine various issues, both organizational and technical, on network interconnection (Cotton, 1977; Sanders and Vinton, 1976; Doll, 1974; Cerf and Kitstein, 1978; Liu, 1978). While previous studies have proposed many solutions, many important issues still remain open. Some of the important ones, either resolved or unresolved, are examined in this section.

2.1 Addressing

Addressing is one of the fundamental issues concerning the connectivity in an internetworking environment. Since each constituent network (subnetwork) may implement some specific naming scheme for identifying sources and destinations of data traffic, the address transformation for internetwork traffic can become very complex. In general, the addressing schemes can be categorized into three categories: unique, composite and hierarchical addresses (McConnell, 1988).

- Unique addressing - Each system in the network is assigned a unique address. The advantage is obviously the easy resolution of address ambiguity. However, to accommodate a potentially large number of systems (stations or nodes), a large address space is needed. Particularly, when a large size address is included in each data packet header, it would result in a waste of the transmission bandwidth. Nevertheless, unique addressing has the advantage of making an attached system independent of the rest of the subnetworks. As a result, a system can be moved among subnetworks without affecting the rest of them. An example of unique addressing is the NSAP-address in the OSI protocol reference model, which is a network-wide address used to identify the network service users in an end system.
- Composite addressing - Composite addressing adds internetwork information to the regular address used in a subnetwork. For example, the subnetwork name can be added as part of the address. When constituent subnetworks use a unique addressing scheme, adding a unique subnetwork name to an address can make an internet address unique. As in the unique addressing scheme, a composite address enlarges the address space.
- Hierarchical addressing - Hierarchical addressing is an extension of the composite addressing scheme. When the constituent subnetworks form a simple hierarchical tree, member subnetworks are nested inside of others and are not visible until the proper higher-level subnetwork is reached. This addressing scheme takes advantages of the internetwork hierarchy to form addresses in a similar nested manner. In this hierarchical structure of addresses, higher-level subnetworks are the backbones of the subnetworks they contain. It can tolerate more duplicated system addresses in different subnetworks.

Generally, the above addressing schemes can be used jointly. For example, basic addressing procedures for interconnecting networks under control of a single organization have been used in (Boggs et al., 1980; Cerf and Kitstein, 1978; Postel, 1980; Haltzer et al., 1981). In such an environment, the same procedures can be uniformly employed throughout all subnetworks. In some typical networks the whole addressing scheme is hierarchical, resulting directly from the definitions of protocol layers. In some other cases, only the lower layers of protocols use consistent addressing in a heterogeneous networking environment.

While some of the basic problems have been solved, a large set of secondary addressing issues still need to be concerned. Examples of such secondary issues are, especially, those dealing with heterogeneous networks under control of different organizations, in such situations, different address schemes are used in different subnetworks and uniform addressing becomes impractical. In (Sunshine, 1982), many of such secondary but advanced issues are discussed; network partioning, multihoming, mobile hosts, and shared access are a few of them. Potential solutions are also proposed for some of the issues. For example, it was discussed that in an interconnected network, if network partitions may occur frequently, some means of forcing use of any available link is desirable (Cerf, 1979). Alternatively, complex methods of updating each network's view of the overall topology, and promulgated knowledge of a partition in one subnetwork to all the others (Periman, 1980; Periman, 1982) can be used. In *Telenet* (Roberts, 1975; Weir et al., 1980), a backup-and-try-alternate method is used.

Other than the low level addressing scheme, at higher levels (layers), an addressing strategy based on the telephone network has been considered by CCITT (X.121 (CCITT, 1978)). Up to 14 digits can be used in its address. The first 4 digits are the destination network identification code (DNIC); the remaining ten digits may be used to implement a hierarchical addressing structure. This scheme does not take into account the need for addressing private networks. As a result, the private networks, under this addressing procedure, appear to be a collection of one or more terminals or host computers on one or more public networks. Also at higher protocol levels, the public carriers have tended to offer services for terminals as well as host access to network facilities. It means that addresses must also be assigned to terminals. Capabilities to distinguish between host and terminal addresses have thus become another issue in this addressing scheme (Cerf and Kitstein, 1978).

2.2 Routing

Routing is the set of functions in charge of choosing a physical, or sometimes logical, path from a source to a destination. When there is only one path between two points (such as star-shaped network), there is no routing needed. In the case of switched networks, there may be several paths between each pair of points and choosing one requires some strategy and associated mechanisms. In particular, when two or more networks are connected, the number of potential physical routes increases. In such a situation, internetwork routing depends on fault detection and congestion control peculiar to each subnetwork. The internetworking facility must be able to coordinate these to adaptively route data between stations on different networks.

To route data frames across networks, a source station/gateway must know if the destination station of a frame is attached to a network to which the gateway is attached or the next gateway along the route to the required destination network if it is not. The major issue with routing, therefore, is how the stations and gateways within the interconnected networks obtain and maintain their routing information.

To obtain routing information within an interconnected network, either a centralized routing or distributed routing scheme can be used. In a centralized routing scheme, the routing information associated with each gateway is downloaded from a central site using special network management messages. The network management system endeavors to maintain their contents up-to-date as networks and stations are added or removed and faults are diagnosed and repaired. This approach is only viable as long as each individual network has its own network management system which incorporates sophisticated configuration and fault management procedures. With a distributed routing scheme, on the other hand, all the stations and gateways cooperate in a distributed way to ensure that the routing information held by each system is up-to-date and consistent. The DoD Internet uses a distributed routing scheme.

Routing tables are generally used to maintained the routing information in each station and gateway. A routing table contains the information that tells, for each possible destination network, the next gateway to which the frames should be forwarded. If every gateway and station in interconnected networks contained a separate entry in its routing table for all other systems, the size of the routing tables and the amount of processing and transmission capability needed to maintain the tables would be excessive and unmanageable. Therefore, the total routing information is commonly organized hierarchically. An example of a hierarchical routing table can be found in the DoD Internet (Comer, 1995).

Routing tables have many drawbacks in a gateway due to gateway's high performance requirement. As reported in (Pitt and Winkler, 1987), with single LAN addresses of 48 bits and LAN's with thousands of stations, table lookup of an address is nontrivial, especially when the lookup time cannot exceed the time it takes to receive a minimum length frame. Even though a routing table can be either programmed into gateways, transmitted to gateways or learned by gateways by observing frames of the network (Hawe et al., 1984; Kummer et al., 1987), the feasibility of routing table is speed-dependent. In addition, if a flat address is used in a routing table, a table with size on the order of half the number of stations in the network is required. If preloading of addresses into bridges is to be avoided and dynamic learning allowed, the flat address constrains the interconnection topology to only spanning trees (Pitt and k. Sy, 1986). On the other hand, if a hierarchical address is used, in which subfields of an address corresponds to a LAN number or region in bridged networks, smaller routing tables can be resulted. However, it would constrain the interconnection topology further (Pitt and k. Sy, 1986), and require a station to obtain a new address when it moves, thereby preventing the use of universally administrated addresses.

To overcome the drawbacks of a routing table, the source routing technique has been examined by many researchers (Sunshine, 1977; Haltzer et al., 1981; Dixon and Pitt, 1988). The source routing technique was developed by IBM for the interconnection of IEEE 802.5 LANs (token Ring) (IEEE, 1985), but it can be applied to other networks as well. In this technique, the source station specifies the route by including a sequential list of gateways in the frame header. On receipt of each frame, a gateway needs only to search the routing field at the frame header for its own identifier. Only if it is present does the gateway forward the frame onto the connected subnetworks. Gateways following the source routing approach do not have to maintain address tables to forward frames; instead, only simple string matching during a linear scan of the routing information stored in the header is performed.

In the source routing approach, a route discovery mechanism is required. A source station can determine the route toward any other station in a number of different ways. The most practical method for route determination is by using a dynamic discovery procedure (Munafo et al., 1993). A station that has no route to a certain target station sends a special broadcast frame to the target station. This broadcast frame is transmitted by gateways in such a way that all the possible routes between the source and the target station (all-route broadcast) are explored, so that the target station receives as many copies of the frame as there are possible routes. When a bridge retransmits one of these route-searching frames, it records in the routing information field of the header the information about the last traversed subnetwork. This dynamic route recording also allows one to recognize and discard copies of the broadcasted frame looping in the interconnected networks. The target station transmits all received copies of the frame back to the source station, routing them along the path recorded in the received header. In this way, the source station receives all the possible routes to the target and stores the best route for future use.

To reduce the amount of traffic generated by the broadcasting frame, an alternative *single-route* broadcast can be used (Munafo et al., 1993). In this alternative scheme, a broadcast frame is sent no more than once per subnetwork. The target station receives a single copy of the frame, and transmits the frame back to the source station along with all the possible routes. The trade-off of this scheme is that the gateways must be configured as a spanning tree in order to avoid frame duplications.

2.3 Flow Control and Congestion Control

Flow control is a procedure through which a pair of communicators regulate traffic flowing from source to destination. Each direction is possibly dealt with separately. A related issue to flow control is congestion control, which is a procedure whereby distributed network resources, such as channel bandwidth, buffer capacity, CPU capacity, and the like are protected from oversubscription by all kinds of network traffic. In general, successful operation of flow-control procedures for every pair of network communicants does not guarantee that the network resources will remain uncongested.

In a multinetwork environment, flow control and congestion control are much more complex problems than those in single network, owing to the possible variations in flow and congestion control policies found in each constituent network. For example, some networks may rigidly control the input of packets into the network and explicitly rule out dropping packets as a means of congestion control. At the other extreme, some networks may drop packets as the sole means of congestion control.

Up to date, very little is known about the behavior of congestion in multiply interconnected networks. It is clear that some mechanisms will be required to permit gateways and networks to assert control over traffic influx, especially when a gateway connects networks of widely varying capacity. This problem is likely to be most visible at gateways joining high-speed local networks to long-haul public networks. The peak rates of the local networks might exceed that of the long-haul networks by factors of 30-100 or more (Cerf and Kitstein, 1978). Generic procedures are needed for gateway/network and gateway/gateway flow and congestion control.

An easy solution is to have gateways to provide buffers to smooth the flow from one network to another (McConnell, 1988; Chou, 1985). Gateways must also send flow control messages to each other so that traffic volume can be modulated to meet differing resource requirements. Eventually, these flow control messages must be passed to the systems generating the traffic in order to shut the flow down at its source.

An alternative approach is to use an allocation technique, whereby a gateway tells other gateways how many more messages it can receive. Allocation can be embedded in traffic being delivered to the other gateways. Thus, gateways have exact knowledge of the amount of traffic they can deliver without flooding the destination. Each new update of allocation information keeps all parties informed of how much they can safely send. Allocation prevents the problem caused by subnetwork delay and can provide a steadier flow. Gateways can allocate more traffic to other gateways that have a higher volume and can change the allocation as condition warrants. Unfortunately, the gateways are still at the mercy of the end-systems unless they are explicitly included as participants (McConnell, 1988).

Yet another approach is to leave the control to the end-system and simply to have the gateways do the best they can. Within the ARPANET environment, gateways provide the end-systems with flow control feedback to elicit their cooperation. In the OSI model, the transport layer within the end-system takes on this task and recovers from lost data caused by overruns.

2.4 Error Control

Error control comprises two major phases: detection and recovery. An error is detected when some communication protocol is violated (e.g. spurious message, number out of range, etc.).

Some errors are deemed unrecoverable, and consequences are to be handled by a higher-layer protocol. There is usually no difficulty in doing a satisfactory mapping of unrecoverable errors of different protocols, to the extent that their occurrences are well identified. Such errors are normally signaled with special messages, which are translated by a gateway.

Error detection and recovery procedures are typically based on acknowledgement and retransmission from a point in past traffic where no error had yet occurred. Thus the two ends of an ongoing data stream must be able to backtrack up to the same safe restarting point. Most protocols use numbering schemes for acknowledgement and backtracking (i.e. message number, packet number, character count, etc.) (Chou, 1985). This is where difficulties arise, because numbering schemes in different networks may not have a well-defined relationship.

A common data structure can be superimposed on top of existing mismatched protocols. This is usually possible, because higher layers handle information units such as a page, transaction, and file. Thus error detection and recovery are pushed into a higher layer, where both ends of a data stream work on the same information quantum, and operate end to end (Schoch, 1979)

2.5 Other Issues

There are still many other issues in network interconnection considered by various researchers (Cerf and Kahn, 1974; McConnell, 1988; Chou, 1985). Some of them are described in the rest of this section. Although these issues are not discussed in great details, they are not necessarily less critical or important than those discussed in the previous sub-sections.

2.5.1 Reliability

The internetwork environment should present a stable service to the using systems. However, its stability depends on a variety of subnetworks that have different degrees of integrity, security, and reliability. Internetwork users may not be able to determine the operational characteristics of all subnetworks, and some may not even be reliable enough to meet the needs of a particular application. Therefore, it is desirable to have the internetworking mechanism resolve incompatibility and to support a close to uniformly reliable service across all subnetworks.

2.5.2 Service

A collection of subnetworks will most likely have incompatible services. Consider a local area network with a connectionless service that is interacting with a connection-based subnet. It is difficult for a gateway to support a consistent internetwork service across these two networks, since their services cannot be reconciled at the same layer.

Another service issue concerns negotiable features, such as priority levels, special message sizes, and the different flow control windows that some subnetworks have. A system can tailor the subnetwork service to suit specific needs when using the service for intranetwork activities, but there may be difficulties in using the same application in the internetwork environment, since other subnets may not have the same set of assumed features. Also, designing internetwork applications is complicated because the wide range of subnetwork facilities makes the calculation of delivery times and data throughput very difficult. Often, the exact environment will be different for each organization.

In general, higher layer procedures (such as Transport, in the OSI model) are needed to handle the incompatibilities in both of the aforementioned situations.

2.5.3 Security

Traffic leaving a subnetwork travels through unknown facilities that are under the control of the other organizations. Therefore, the end-system must take major responsibility for the security of internetwork communications. The necessary protection can range from preventing interception or alteration to detecting bogus traffic. The security of interconnected subnetworks is only as strong as that of the weakest member.

2.5.4 Information Quantum

Information exchanged between layers of protocols is always structured. Units exchanged are not just bits, but packets, messages, blocks, items, etc. Control mechanisms operate control fields, some of which contain numbers used to identify uniquely a particular piece of information. Numbers are typically assigned sequently, and they are essential for error and flow control mechanisms.

As discussed in Section 2.3 and Section 2.4, numbers must be translated when mapping a protocol into another. There is no simple algorithm; e.g. one protocol may handle message numbers, and the other packet numbers. The relationship would be simple if messages always contained the same number of packets. In most cases lengths are variable, and the relationship between two numbering schemes cannot be predicted accurately.

2.5.5 Dialogue

This mechanism determines whose turn it is to transmit in an interactive communication session. Modern systems handle this matter in a formal manner, by placing delimiters, or special flags, in the data stream. The initial turn is determined by convention, or by contention. Additional features are made available to speed up or force a turn exchange. These conventions constitute a dialogue protocol, which may be built into an application-oriented protocol, such as virtual terminal, file transfer, remote job entry, and so on.

Older systems do not generally use such conventions. Turn exchange is determined as a result of message parsing, special character sequences, or as an established custom. All that is highly context sensitive, and in some cases ambiguous.

In an environment of network interconnection, mapping a fuzzy turn system into a formal one is generally not feasible. A typical way around this consists of setting a free mode, in which both ends may transmit at any time. The users are then warned to follow specific conventions, lest they get into unpredictable traps.

2.5.6 Administration and Accounting

Administrative issues should be considered very early in internetworking design. Interconnected networks must have accounting mechanisms, especially when many of them interconnect public data networks. Technically, accounting (i.e. gathering billing information) could be viewed as an appendage to supervision. A minimum of consistency across networks is required, so that traffic exchanged can be accounted for in the same manner on each side of the border. Some networks bill characters, others bill packets. There can only be ad hoc solutions to specific problems.

3 Approaches to Network Interconnection

From a user's viewpoint, an interconnected set of networks is not different from a single network. It only provides an enlarged population of users with enlarged services, and this simple view must be preserved. In other words, users should not be bothered by interconnection problems.

From an internal point of view, an interconnected set of networks is not necessarily different from a single network. In particular, two identical networks (e.g. from the same computer manufacturer) can usually be integrated into a single bigger one. However, in many cases, networks to be interconnected are not identical. They are usually designed in different environments, at different periods, with different constraints and technologies, etc. In addition, it is essential to preserve freedom in the design of future computer networks and still be able to interconnect them with existing ones. In other words, the question is how to interconnect heterogeneous networks rather than how to build a world wide homogeneous network.

Nevertheless, homogeneous network interconnection is commonly used due to its simplicity. In this section, techniques to interconnect networks are categorized into homogeneous network interconnection and heterogeneous network interconnection. Various approaches in each category are discussed.

3.1 Homogeneous Network Interconnection

In a homogeneous network environment, the effect of internetworking is to enlarge the coverage of each network and a gateway/bridge serves as the point of traffic relay. Since the same protocols are used, there is no need to perform protocol conversion between networks.

3.1.1 Bridging Individual Layers

In Fig. 7, an intermediate node bridge at layer N of two homogeneous networks is shown. In this

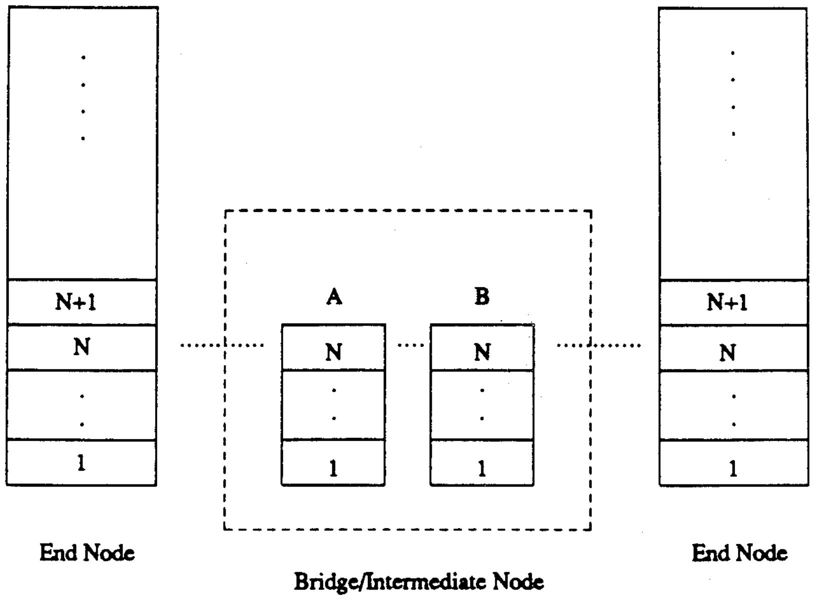


Figure 7: Bridging Individual Layer

approach, an end-to-end internetwork connection service is supported (Stallings, 1994): the transmission across multiple networks requires a common end-to-end protocol for providing reliable end-to-end service. This simplest form of interconnection fits neatly in a configuration of multiple homogeneous networks. In Fig. 7, the two halves (A and B) of the bridge can be physically separated. If they are

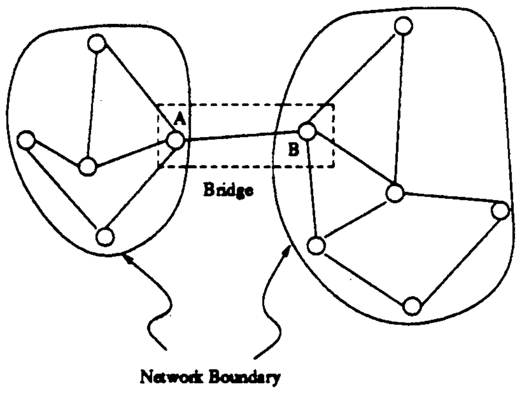


Figure 8: Separated Bridge

separated, each one of them belongs to one of the opposite sub-networks that the bridge connects as shown in Fig. 8. No new additional software is needed at their protocol stacks. The two halves of the bridge, A and B, do need extra routing information, however, to know when to relay internetwork traffic to the other network.

On the other hand, if the two halves of the bridge are put into the same node, an additional adaptor layer on top of the bridged layer must be used to perform the traffic hand-over function as described in the next subsection.

3.1.2 Adding Internet Layers

Fig. 9 shows the configuration of the networks when the two halves are put into a single bridge node.

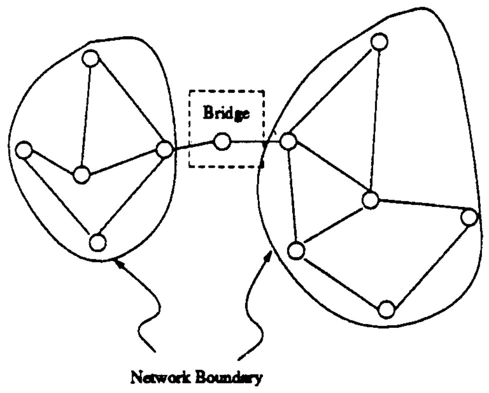


Figure 9: A Single Node Bridge

In such a configuration, an adaptor layers in the protocol stack is needed as shown in Fig. 10. This

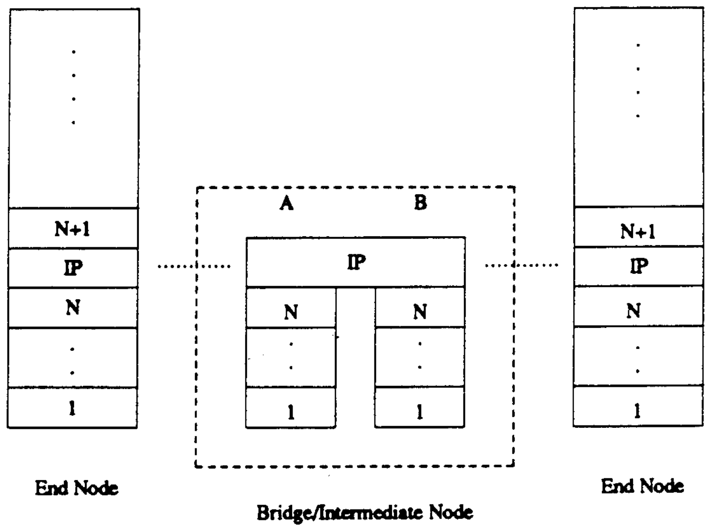


Figure 10: Adding Internet Layer

mechanism is used by the DARPA Internet to interconnect multiple local area networks. It uses an added protocol above the network layer, called *Internet Protocol* (IP). DoD IP has been standardized by the Department of Defense. A similar standard, ISO IP, has also been developed within ISO. More discussion on IP (DoD and ISO) is provided in Section 4.1 and Section 4.2.

Similar to IP, X.75 is another protocol used as an added layer to perform the internetworking functions. Unlike IP, which supports reliable end-to-end transmission service, X.75 relies upon each sub-network to provide reliable service within themselves and splices together individual network connections across multiple networks. This type of interconnections is called *network-by-network* approach (Stallings, 1994). X.75 is designed as an extension of X.25. To two stations on different networks, it appears that they have a single virtual circuit connecting them. In fact, the virtual circuits terminate at the node gateways, which maintain the status information required to connect separate virtual circuits. Since X.75 requires all sub-networks to be X.25, it is less flexible than DoD IP which places no significant restrictions on internal network protocols.

3.2 Heterogeneous Network Interconnection

In (Gien and Zimmermann, 1979), it was shown that the technologies that were used to interconnect heterogeneous computers can be adopted, with slight adaptation, to interconnect heterogeneous networks. It is clear that the heterogeneous network interconnection problem boils down to the degree of *similarity* between the interconnected networks; or, the *difference* of the protocols used in these interconnected networks. *Protocol mismatch* (Green, 1986) is the single most important issue needs to be solved in heterogeneous network interconnections. The approaches to solving the mismatch problem are categorized into *layer substitution* and *intermediate converter* in the following sub-sections.

3.2.1 Layer Substitution

Layer substitution is a practical solution in typical situations such as replacing a private packet network layer by a public packet network layer, while all other components remain unchanged. The network model for layer substitution is shown in Fig. 11, where layer A_n of network A is replaced by B_n. In this model, it is assumed that all other layers are compatible between networks A and B. Only layer n has protocol mismatches.

As long as the service provided by A_n is *similar* to the one provided by B_n, it is feasible to map interface of A_n onto that of B_n through an *interface adaptor*. There is no good measure of similarity between A_n and B_n. The complexity of the interface adaptor and the characteristics of the service should be assessed.

When the original and the new service are functionally very different, it is not always clear that a substitution is possible without affecting higher layers. For example, replacing a transmission system using leased or switched circuits by a packet network may require the replacement of higher layers, because transit delays deteriorate their performance.

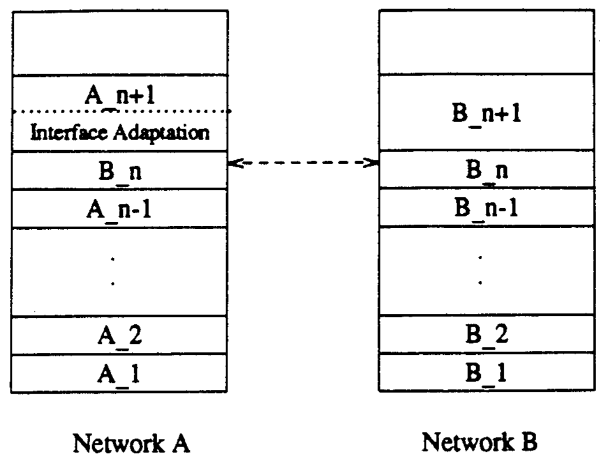


Figure 11: Layer Substitution

In principle, layering is intended to prevent the propagation of modification throughout the entire system. However, this is effective only to the extent that that potential future characteristics have been more or less anticipated. Occasionally, new technologies based on new concepts require major changes in the overall system architecture.

In terms of network configuration, this approach does not use an intermediate node since the service by the substituting layer, B_n, is provided directly to layers A_{n+1} in the same node. Thus, its configuration is the same as that shown in Fig. 8.

3.2.2 Intermediate Converters

A more common conversion configuration is as shown in Fig. 9. The corresponding network model is shown in Fig. 12. In this model, the protocol P of network A and protocol Q of network B are broken down into sender and receiver entities; i.e. P_a and P_b for protocol P and Q_a and Q_b for protocol Q. The entities, P_b and Q_a, are combined in the intermediate converter node. At the network boundary, the end nodes do not need to be aware of the other network since the converter will hide all the protocol mismatches from them. This *transparency* property is essential in heterogeneous network interconnection to prevent modification from propagating throughout all the system.

Compared with the layer substitution model, this intermediate node model is more transparent than the end nodes model, since no changes in the end node protocol stack is needed. In addition, it places less restrictions on the system; e.g. it makes no assumption about the layered protocol structure. On the other hand, due to its high degree of abstraction and generalization, the intermediate converter model usually is more complex to apply. Also, due to the intermediate converter node, extra transit delay may be introduced, which renders its performance lower than that of the end node conversion such as the layer substitution model.

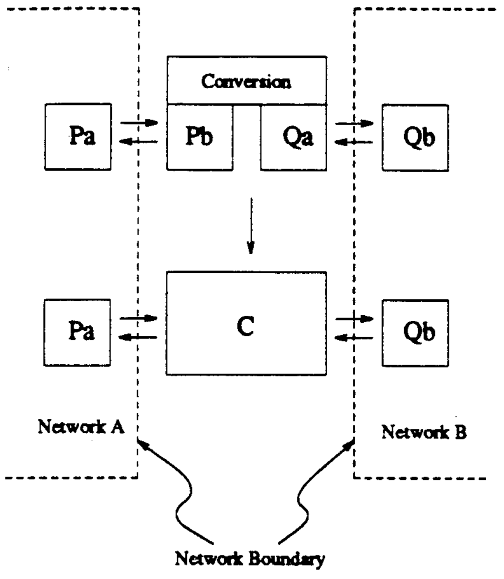


Figure 12: Intermediate Converter

Many ad hoc protocol conversion algorithms use this intermediate conversion node model (Francois and Potocki, 1983; Groenback, 1986; Rodriguez, 1988; Rose and Cass, 1987). Since the ad hoc conversion algorithms cannot be applied to protocols other than their original targets, they are proven to be very expensive and limited. On the other hand, following Green's pioneering work in (Green, 1986), many algorithms based on formal methods have been proposed in the past decade. It is believed that to solve the protocol conversion problem efficiently and once for all, the formal method should be used. Most of the formal methods use this intermediate converter model. The detail of protocol conversion formal methods is provided in the second part of this article (Section 6 to Section 9).

4 Interconnection of Low to Medium Speed Networks

4.1 ISO Standard

Multiple X.25 WANs can be interconnected by gateways based on X.75 (Halsall, 1992), which is essentially a connection-oriented protocol. On the other hand, X.25 can operate in either a connection-oriented mode or in a pseudo-connectionless mode by using fast select and its packet-layer protocol can be used to interconnect LANs as an internet-wide protocol. In mid-1980s, ISO adopted the internetwork protocol standard, ISO 8473, which provides connectionless service (from X.25). This connectionless service contrasts with the connection-oriented X.75 service. There are a number of advantages to the connectionless approach (Stallings, 1994):

- A connectionless internet facility is flexible. It can deal with a variety of networks, some of which are themselves connectionless.
- A connectionless internet service can be made highly robust. This is basically the same argument made for a datagram network service versus a virtual circuit service.
- The connectionless internet service is best for connectionless transport protocols.

A model of using ISO IP in internetworking LANs is shown in Fig. 13. Using ISO IP protocol,

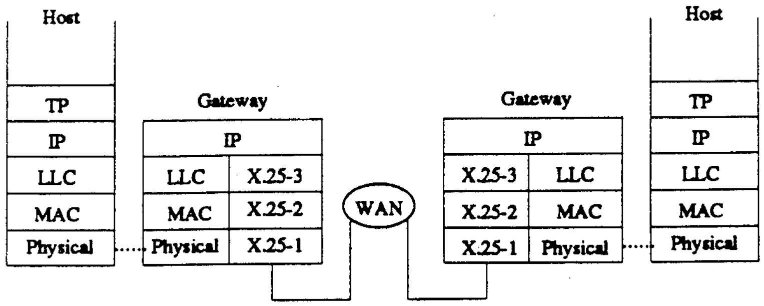


Figure 13: ISO Internet Protocol Model

the IP user requests transmission of a unit of data with N-UNITDATA.request user primitive. N-UNITDATA.indication is then used by IP to notify a user of the arrival of a data unit. The gateways can discard internetwork traffic in a variety of circumstances, such as transmission or format errors, congestion, or excessive delivery delay. It allows the sending system to receive an error report. The nature of a connectionless service requires a transport layer to organize internet traffic into reliable streams of PDUs.

ISO IP uses the addressing scheme specified in ISO 8348, Addendum 2. The self-describing feature of a connectionless PDU allows any gateway to interpret the header easily: gateways receive internetwork PDUs from an attached subnetwork; after verifying the internetwork header, they use the internetwork information as a guide in processing the internetwork PDU. A major gateway operation is the selection of a next relay destination. Information included in the internet PDU and locally maintained routing data are the sources of information for the gateways. After a relay is determined, the internetwork PDU is encapsulated within another subnetwork PDU and sent to the next relay point of the end system. Segmentation may be used if the new subnetwork has restrictive size limitations. Subnetwork-specific addressing and framing are added to the enclosing subnet PDU.

The PDU header is the fixed part of the internetwork PDU. The highlight of the ISO IP PDU format is shown in Fig. 14. The *Type* field indicates whether this is a Data (contains user data) or Error (contains error report) PDU. The address part of the PDU is mandatory. The source and destination addresses are variable-length fields, and the source address follows the destination address. Each internetwork address is preceded by an octet that tells its length (ISO 8348, Addendum 2, specifies the addressing structures). The *Segment Offset* is used when the original data is segmented into several PDUs due to the length restriction in subnetworks; it tells the relative position of a

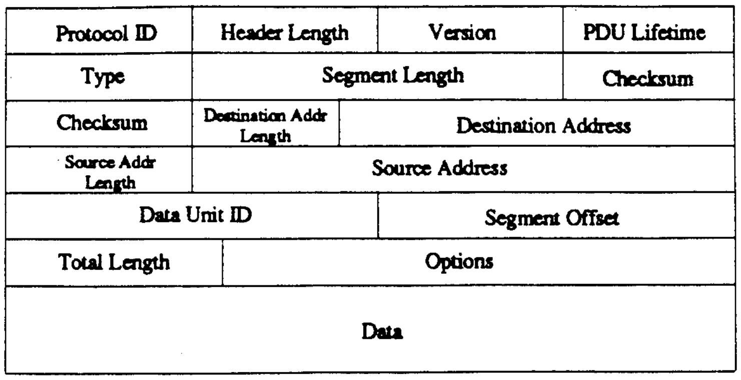


Figure 14: ISO IP PDU Header

segment in the data field of the reassembled PDU. Reassembly is intended to be performed at the destination. Reassembly at an intermediate point is not forbidden, however. The *Data Unit Id* is assigned by the sender and stays with all segments, even if they undergo subsequent segmentation. During the segmentation, the offset is adjusted accordingly.

In addition to ISO 8473, ISO IP is also based on an associated connectionless SNICP (Subnetwork Independent Convergence Protocol), which is defined in ISO 8475. In general, ISO 8475 is an internet-wide, connectionless, subnetwork-independent convergence protocol. Besides a full set of internetworking protocols, there are two other sets: the *inactive network layer protocol* and the *non-segmenting protocol*.

The inactive network layer protocol is the connectionless protocol often used with LANs and is intended for applications involving a single network. Thus the source and destination end systems are both connected to the same network so none of the harmonizing functions (such as re-encapsulation) are required (Halsall, 1992). The non-segmenting protocol is intended for use in internets that consist of subnetworks which have the maximum packet size less than or equal to that of a single internet protocol data unit (NS-user data unit - NSDU). Clearly, the segmentation function associated with the protocol is not required in this case.

In practice, ISO IP is based on the internet protocol that has been developed as part of research into internetworking funded by the U.S. Defense Advanced Research Project Agency (DARPA). DoD IP is described in the next sub-section.

4.2 DoD Standard

The early DARPA internet, ARPANET, was used to interconnect the computer networks associated with a small number of research and university sites with those of DARPA. When it came into being in the early 1970s, ARPANET involved just a small number of networks and associated host computers. Since then, the internet has grown steadily. Instead of just a number of mainframe computers at each site, there are now large numbers of worksations. Moreover, the introduction of LANs means that there are now several thousand networks/subnets. ARPANET is now linked to other internets. The combined internet, which is jointly funded by a number of agencies, is thus known simply as *the Internet.*

The *Internet Protocol* (IP) is the name given to the protocol standard developed by DoD (Department of Defense) as part of the DARPA internet project. Other than IP, the Internet also includes transport and application protocols which are now used as the basis of many other commercial and research networks. The complete protocol suite used in the Internet is known as *TCP/IP* (Transmission Control Protocol/Internet Protocol). As discussed in the previous sub-section, ISO IP is specified after DoD IP. Consequently, the general protocol model of DoD IP is similar to that of ISO IP shown in Fig. 13, with the exception that the lower layers of the Internet do not have to be the X.25 protocols.

In DoD terminology, each protocol data unit transmitted across the Internet is called *datagram.* The Internet datagram format is shown in Fig. 15. The *type of service* plays the same role of

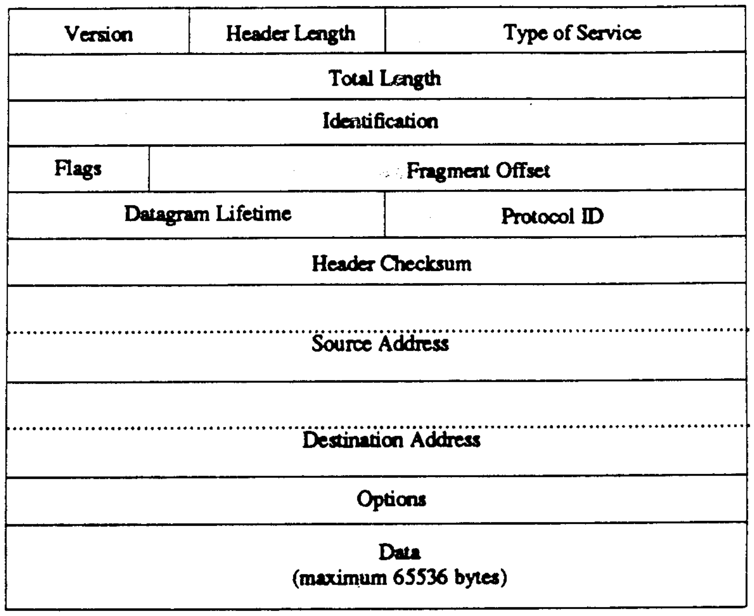


Figure 15: The Internet Datagram Header

the quality of service parameter used in ISO networks. The *identification* field is used to allow a destination host to relate different datagrams to the same user message. The *flags* and *fragment offset* are used when original user data need to be fragmented into smaller datagrams during transmission. The *protocol* field is used to enable the destination IP to pass the datagram to the required protocol since there are more than one protocol associated with the TCP/IP protocol suite. Finally, the *options* field is used in some datagrams for functions such as error reporting, debugging and route redirection.

Like ISO IP, DoD IP is an internet-wide protocol. It enables two transport protocol entities resident in different end systems/hosts to exchange message units (NSDUs) in a transparent way. This means that the presence of multiple, possibly different, networks/subnets and intermediate systems/gateways is completely transparent to both communicating transport entities. As DoD IP is a connectionless protocol, message units (datagrams) are transferred using an unacknowledged best-try approach.

The network services provided by the Internet can be classified as either application level internet service or network level internet service (Comer, 1995). From the view point of a user, the Internet appears to be a set of application programs that use the network to carry out useful communication tasks. Most users that access the Internet by invoking application programs without understanding the Internet technology or even the path their data travels to its destination; they rely on the application programs to handle such details. Only programmers who write such application programs view the Internet as a network and need to understand the details of the technology. The most popular and widespread Internet application programs include: Electronic mail, file transfer and remote login.

At the network level, the Internet provides two broad types of service that all application programs use. They include connectionless packet delivery service and reliable stream transport service.

The connectionless packet delivery service forms the basis for all other Internet services. This service routes small messages from one machine to another based on address information carried in the message. Because the connectionless service routes each packet separately, it does not guarantee reliable delivery. Also because it usually maps directly onto the underlying hardware, the connectionless service is extremely efficient. More importantly, this service makes the Internet protocols adaptable to a wide range of network hardware.

The reliable stream transport service is needed because most applications require the Internet to recover from transmission errors, lost packets, or failures of intermediate machines along the path. With the stream transport service, an application on one computer is allowed to establish a *connection* with an application on another to send a large volume of data across as if it were a permanent, direct hardware connection. Underneath, of course, the Internet protocols divide the stream of data into small messages and send them, one at a time, waiting for the receiver to acknowledge reception.

As reported in (Comer, 1995), the characteristics of the Internet services are described as follows:

- Network technology independent - While the Internet is based on conventional packet switching technology, it is independent of any particular vendor's hardware. Most importantly, the Internet includes a variety of network technologies ranging from networks designed to operate within a single building to those designed to span large distances.
- Universal interconnection - The Internet allows any pair of computers to which it attaches to communicate. Each computer is assigned an *address* that is universally recognized throughout

the Internet. Each datagram carries the addresses of its source and destination. The destination address is used to make routing decisions.

- End-to-End acknowledgements - The Internet protocols provide acknowledgements between the source and ultimate destination instead of between successive machines along the path, even when the two machines do not connect to a common physical network.
- Application program standards - In addition to the basic transport-level services (like reliable stream connections), the Internet includes standards for many common applications. Thus, when designing application programs that use the Internet, programmers often find that existing software provides the communication services they need.

In terms of internetworking with dissimilar networks, the DoD IP protocol provides a number of core functions and associated procedures to carry out the various harmonizing functions:

- Fragmentation and reassembly - This is concerned with issues relating to the transfer of user messages across subnetworks which support smaller packet sizes than the user data.
- Routing - To perform the routing function, the IP in each source host must know the location of the internet gateway or local router that is attached to the same network. Also, the IP in each gateway must know the route to be followed to each other subnetworks.
- Error reporting - When routing or reassembling datagrams within a host or gateway, the IP may discard some datagrams. This function is concerned with reporting such occurrences back to the IP in the source host and with a number of other reporting functions.

DoD IP has been in use since the mid-1970s and is available in many types of equipment. All the TCP/IP protocol specifications are publicly available, as a result of which the Internet is by far the largest currently operational internet based on an open standard.

4.3 Conversion Between TCP and ISO Transport Protocol

Since TCP/IP was adopted as official DoD internetworking standard in 1978, it has become more and more popular and been broadly used in both military and civilian applications for many years. In the meantime, the need for the growing ISO community to interoperate with the existing TCP community has also been increasing steadily. In general, the TCP standard corresponds roughly to the ISO Class 4 Transport Protocol. Thus, the conversion between TCP and the ISO Class 4 Transport Protocol has been considered as a reasonable step toward the interoperability between the ISO and TCP protocols. In (Groenback, 1986), the feasibility of such conversion is examined and the result is introduced in this sub-section.

The conversion model proposed in (Groenback, 1986), in the context of the OSI basic reference model, is shown in Fig. 16. The notations used in Fig. 16 are explained as follows:

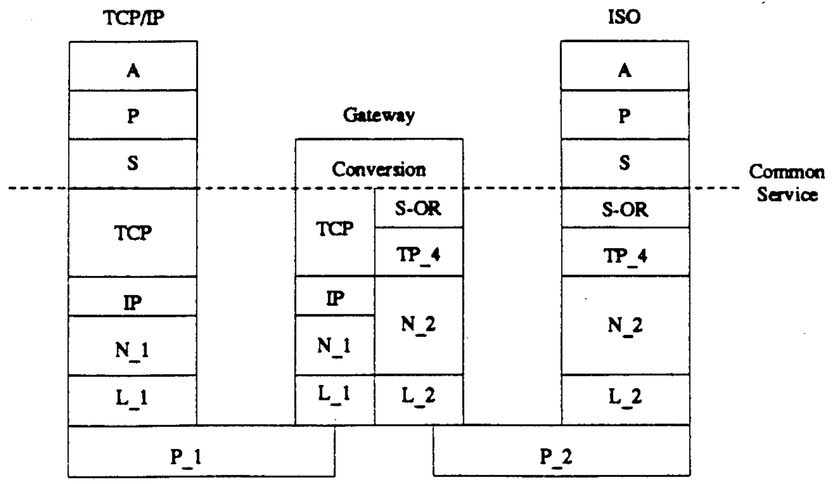


Figure 16: TCP - ISO Conversion Model

- A - Application layer protocols (common to all systems)
- P - Presentation layer protocols (common to all systems)
- S - ISO session sub-layer protocols excluding orderly release (common to all systems)
- S-OR - ISO session orderly release sub-layer
- TP_4 - ISO class 4 transport protocol
- TCP - DoD transmission control protocol
- IP - DoD internet protocol
- N_1 - DoD subnetwork protocol
- N_2 - ISO network layer
- L_1 - DoD link layer
- L_2 - ISO link layer
- P_1 - DoD physical layer
- P_2 - ISO physical layer

The strategy adopted in this internetwork conversion example is to compare the service provided by TCP to that provided by ISO protocols (Groenback, 1984), and then sketch a conversion procedure which retains the end-to-end significance of the protocols. The similarity of the two transport protocols makes it possible to achieve this with only minor restrictions to the service supported. The common subset for which the conversion takes place includes the ISO Session Orderly Release, and appears sufficiently powerful for most applications.

Interoperability over the subnetwork boundaries is not required in Layers 1-3 (inclusive) of the OSI model. This is because these layers deal with node-to-node and intranetwork functions. The main exception to this is the global addressing which functionally belongs to the Internet sub-layer of the Network layer. The implementation of the Internet sub-layer is not generally mandatory (although it is a requirement in the U.S. DoD environment), but an internet address must be carried across the different subnetworks, either by the Network layer or the Transport layer.

Note that in Fig. 16, it is shown that every host has the same protocols in all layers above the Network layer to achieve host-level interoperability. Without such assumption, conversion would be quite impossible. It is discussed in (Groenback, 1986) that without common high level protocols, the conversion has to be done for all layers. However, the higher protocol layers (those above the Transport layer) are so diverse that an all-embracing protocol conversion which retains the end-to-end conditions for every layer is not feasible. Moreover, even though such an all layer converter can be constructed, it would act as an intermediate store-and-forward unit due to its complexity and conversion delay introduced. This implies that the store-and-forward unit would have to be responsible for delivery of data to the real end systems. Such an arrangement would not be suitable for certain applications in which survivability and delay are critical.

Therefore, the conversion which retains the end-to-end operation must be without intermediate thrusted relay, and this should take place at a layer for which it is possible to establish the end-to-end conditions for all subsequent lower layers. In this example, the conversion is performed at the Transport layer which maintains the end-to-end conditions for a common subset of the services provided by the TCP and ISO protocols.

Interested reader should refer to (Groenback, 1986) for the conversion algorithm, which is in the form of an extended finite-state machine (EFSM) with descriptions of transitions and actions for every state/event combination. Finally, it was claimed that the algorithm is no more complex than that of the more complicated of TCP and ISO protocols due to a great deal of similarity between the two protocols and the power of common service subset.

4.4 SNA Network Interconnection

Systems Network Architecture (SNA) allows terminals and application programs to communicate with one another. This network was introduced in 1974 by IBM and its functions have been continuously enhanced. While the intranetwork capabilities of the SNA networks were enhanced during its early years (Sundstrom and Schultz, 1980), there was also a growing need to interconnecting multiple SNA networks to allow broader coverage. In (Benjamin et al., 1983), J. Benjamin *et al.* introduced the SNA interconnection functions. The approach adopted by them is a typical example of homogeneous network interconnection, which supports transparent interconnections among users on different networks; i.e. a communication user need not be aware that a session partner is in a separate network. In this sub-section, the basic SNA concepts are briefly reviewed and then homogeneous SNA network interconnections are discussed based on the solution proposed in (Benjamin et al., 1983).

Essentially, SNA defines logical entities that are related to the physical entities in a network and specifies the rules for interaction among those logical entities. The logical entities of an SNA network include network addressable units and the path control network that connects them. Network addressable units communicate with one another using logical connections called *sessions*, as shown in Fig. 17. The three types of *Network Addressable Units* (NAUs) are the Logical Unit (LU), the

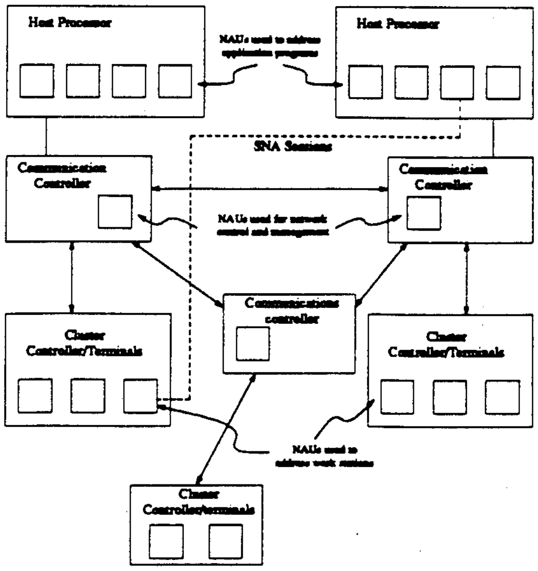

Host Processor
NAUs used to address application programs
Host Processor
SNA Sessions
Communication Controller
Communication Controller
NAUs used for network control and management
Communications controller
Cluster Controller/Terminals
Cluster Controller/Terminals
NAUs used to address work stations
Cluster Controller/terminals

Figure 17: SNA Network

Physical Unit (PU), and the System Service Control Point (SSCP):

- LU - An LU is a port through which end users may access the SNA network. An end user can use an LU to communicate with one another and request services of an SSCP.
- PU - A PU is a component that manages the resources of a node in cooperation with an SSCP.
- SSCP - This is a focal point for configuration management, problem determination, and directory services for end users.

Each message sent to a network addressable unit is prefixed by a transmission header, which includes address of the NAU. The address consists of two parts, the *subarea* field and the *element* field. Each major node in the network is defined as a *subarea* node. The subarea field of the network address is used to route a message in the path control network from origin subarea to destination subarea, possibly through some intermediate subareas. The destination subarea then delivers the message to the appropriate network addressable unit by using the element field of the address.

In addition to a network address, each NAU has a *network name*. A network name is a symbolic identifier used to refer to a NAU. Thus, each network has a *name space* and an *address space*. An

SSCP directory service consists of mapping names to addresses for those NAUs in the SSCP's domain of control. A cooperative directory service is provided within each SSCP in the network to resolve mapping of names to addresses between domains, so that a directory entry at one SSCP points to the SSCP that can provide the resolution.

In the SNA network, when a session is set up between end users in different subareas, a particular physical path through the network's subareas and links is determined. The selection of the path is made indirectly through the specification of a *class-of-service* name. This symbolic name designates desired communication characteristics, such as path security, transmission priority, or bandwidth. The class-of-service name is mapped to a list of *virtual routes*, any one of which can be selected for the session. A virtual route is a logical connection between the two end users' subarea nodes. It supports protocols that provide data integrity, network transmission priority, and flow control functions.

A virtual route is itself mapped to a set of links and nodes called an *explicit route*. The explicit route is the physical path that is used by the session. An explicit route may be shared by multiple virtual routes. In addition, each virtual route can be shared by multiple sessions.

Most SNA networks evolve with different address-splits and user-selected logical-unit names. For traffic to flow between the networks, route definitions are required, which would not be possible if there are ambiguities caused by duplicate addresses; i.e. the duplication of LU names would make session initiation request ambiguous. Therefore, in this typical example of homogeneous SNA network interconnection, a gateway between networks to translate names and addresses is necessary to resolve the naming ambiguity. Viewed from any one of the interconnected networks, the gateway is a part of that network. In addition, the gateway's participation in other networks need not be known to a given network. The address and name translation are performed at the gateway and are transparent to the networks it connects. More specifically, this translation is performed on the addresses in the transmission header of all messages sent on internetwork sessions. LUs and SSCPs in one network are given alias addresses in the gateway to allow their representation in another network. Thus, routes are independently defined in each network, and the gateway serves as a virtual route end point. In this example, to prevents one network from taking control of another network's resources, PUs are not represented by alias addresses in the gateway.

The use of alias addresses in the gateway to represent LUs and SSCPs in other networks limits the number of such resources that can be concurrently addressed. This limit is the number of element addresses in the gateway subarea. This limitation is achieved through dynamic use of the element addresses in the gateway subarea. At session termination, when there is no further need for the address, an alias name is returned to the free pool so that it may subsequently represent a different network addressable unit. Thus, the number of successively addressable NAUs is not limited. The apparent number of terminals or application programs accessible to each end user increases above what is normally available within one SNA network.

Alias addresses allow communication with NAUs in other networks as though they were part of the same network. Similarly, alias names can be used to represent other network names to avoid ambiguities that would otherwise result from duplicate names in the networks. The address translation is done by the gateway at the path control layer of the SNA protocol. Since it does not affect the session protocols, network independency and autonomy are preserved with this approach.

4.5 Conversion Between OSI and SNA

In the previous sub-section, an approach to interconnecting homogeneous SNA networks was introduced. In this sub-section, an example of heterogeneous interconnection of SNA with an OSI network is discussed.

As OSI becomes an international standard, it is gaining support in both industry and government agencies since one of the major applications of OSI is to act as an intermediary between heterogeneous networks. By now, OSI is a reality and, the growth in its market acceptance during the recent years has been reinforced by the increasing availability of OSI products from a wide spectrum of computer manufacturers. In (Sy et al., 1987), the interconnection of OSI networks with SNA networks is discussed. Also explored are fundamental relationships between SNA Logical Unit (LU) type 6.2 and OSI session and transport layers.

As discussed in the previous sub-section, LU is a port into the SNA network through which an end-user can communicate with other end-users. There are several LU types, differing in the specific SNA protocols and options supported. LU type 1 is for a session between an application program and a terminal. LU type 2 is for a session between an application program and a display using the IBM 3270 data stream. LU type 3 is for a session between an application program and a printer using the IBM 3270 data stream. LU type 4 is for a session between an application program and a terminal or a session between two terminals. LU type 6 is for a session between two application programs.

LU type 6.2 is a relatively later addition to the SNA architecture (compared to other SNA LUs). It goes further than previous LU types. In addition to defining a rationalized subset of the protocols, it also (Mairs, 1990):

- provides a distributed operating platform for transaction programs.
- presents a rigorously defined service boundary to transaction programs.
- interfaces with other facilities of the local environment such as database managers, to coordinate synchronization and security.
- provides many performance features and options.
- offers peer-to-peer communication which is conceptually the same as that offered by OSI.
- has a well-defined interface which makes mapping comparatively easier than with the other LU types.

The approach taken in (Sy et al., 1987) to interconnect SNA and OSI networks is to perform protocol conversion between the LU type 6.2 and OSI protocols. To do so, the common functions of SNA and OSI are identified, which include:

- OSI session connection with SNA LU type 6.2 *conversation*.
- OSI transport connection with SNA LU type 6.2 session.

Although the terminologies are different, many of the functions supported are similar. These identified functions, in turn, become the basic mapping between the SNA and OSI architectures. Based upon the function mapping identified, an OSI-SNA gateway/converter can then be constructed. From a very high level, a typical example of using such protocol conversion based upon function

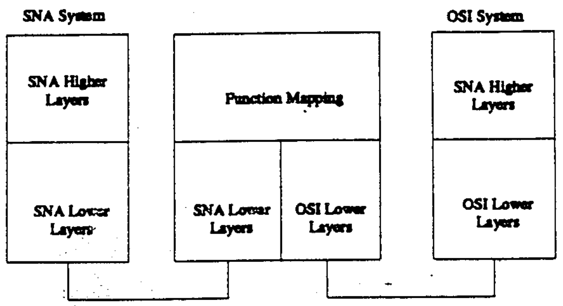


Figure 18: SNA - OSI Protocol Conversion

mapping between SNA and OSI networks is shown in Fig. 18.

Since the conversion between SNA LU type 6.2 and OSI protocols provides a *general* communication path between SNA and OSI networks, it is important to realize that SNA LU type 6.2, in the same way as an OSI protocol layer, is only a platform upon which applications that perform useful functions can be built. It is not in itself a complete facility, but only the plumbing. Some of the applications that reside above LU type 6.2 are provided by IBM and others will be user-written transactions.

In Fig. 18, the same high level protocols are used in both SNA and OSI systems such that only the lower layers need to be converted. It is pointed out in (Mairs, 1990) that it is not always possible to identify the appropriate SNA application to which a gateway should be built. For example, there are many IBM file transfer products (e.g. DSX, RJE, ADCS, SSS, CICS/BDI); it is not clear whether FTAM of OSI can be mapped on to any of them. As a result, most protocol converter/gateway usually convert only the lower layers as the LU-6.2/OSI converter discussed in this example. It implies awareness of the *foreign* architecture in at least one of the networks as shown in Fig. 18.

The conceptual structure of the proposed gateway is shown in Fig. 19. The structure consists of three components:

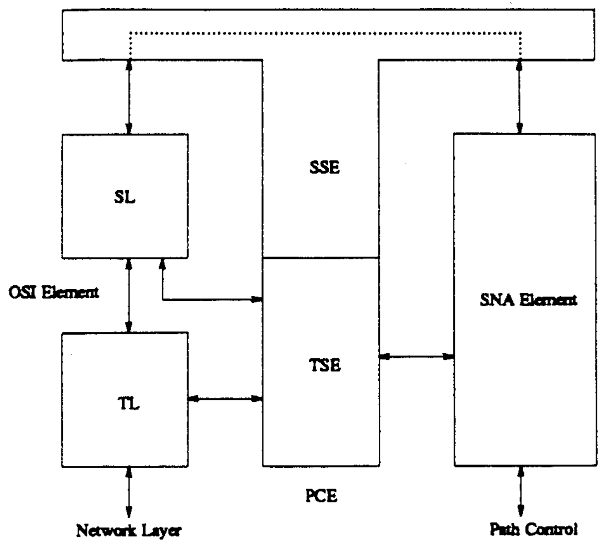


Figure 19: OSI-SNA Gateway Structure

- OSI Element - the portion of the OSI-SNA gateway that communicates with the partner OSI node. It only handles OSI protocol, and is divided into two subcomponents, the Session Layer Sub-Element (OSI-SL) and the Transport Layer Sub-Element (OSI-TL).
- SNA Element - the portion of the OSI-SNA gateway that communicates with the partner SNA node. It only handles SNA LU type 6.2 protocol.
- Protocol Conversion Element (PCE) - the portion of the OSI-SNA gateway that converts OSI protocol to SNA LU type 6.2 protocol and vice versa. It is divided into two subcomponents, the PCE Session Sub-Element (PCE-SSE) and the PCE Transport Sub-Element (PCE-TSE). PCE-SSE converts OSI session protocol to SNA LU type 6.2 protocol and vice versa. PCE-TSE converts OSI Transport protocol to SNA LU-6.2 session activation/deactivation protocol and vice versa.

Between these components, there are six boundaries defined in this OSI-SNA gateway:

- SSE-SL - used for communication between PCE-SSE and OSI-SL and is based on OSI session service primitives.
- SSE-SNA - used for communication between PCE-SSE and SNA element and is based on the SNA LU type 6.2 conversation verbs.
- TSE-TL - used for communication between PCE-TSE and OSI-TL and is a subset of the OSI transport service primitives.
- SL-TL - used for communication between OSI-SL and OSI-TL and is the remaining subset of the OSI transport service primitives.

- TSE-SNA - used for communication between PCE-TSE and the SNA element and deals with the establishment and the release of an SNA LU-6.2 session.
- TSE-SL - used for communication between PCE-TSE and OSI-SL and is a subset of the OSI transport service primitives. This subset deals with the establishment and release of an OSI transport connection.

With such a gateway, when an incoming command related to the establishment of OSI transport connection is received by the gateway from the partner OSI node, it is changed to the appropriate service primitive, which is then forwarded to PCE-TSE. If the transport connection has been established and the command is related to the session of higher layers, the command is changed to the appropriate service primitives, which is forwarded to OSI-SL. OSI-SL then generates the appropriate service primitive, which is forwarded to PCE-SSE.

When the command is from the partner SNA node and is related to the establishment of an SNA LU type 6.2 session, an appropriate service primitive will be forwarded to PCE-TSE. If the SNA LU type 6.2 session has been established and the incoming command is related to a conversation, the appropriate parameters will be forwarded to PCE-SSE.

The gateway design introduced in this sub-section is the result of a joint study between IBM Japan and NTT (Nippon Telegraph and Telephone Corporation) in 1984. Other than the basic operations just discussed, also included in the study are issues such as address translation between SNA and OSI networks, exception handling, etc.

Another example of heterogeneous internetworking with SNA networks is discussed in (Zoline and Lidinsky, 1985), in which a similar protocol conversion approach as described in this sub-section is proposed to interconnect SNA networks with XNS networks, which are vendor proprietary networks from Xerox. Yet another SNA heterogeneous internetworking example is studied in (Wakayama et al., 1985), in which the interconnection of SNA networks with DCNA networks is examined. The approach taken in (Wakayama et al., 1985) is quite unique, which does not perform protocol conversion between SNA and DCNA protocols directly. Instead, the OSI is used as an intermediary protocol and the concepts and functions of both SNA and DCNA networks are mapped onto that of OSI. Interested readers should refer to (Wakayama et al., 1985) for details.

4.6 LAN Interconnection Through MAN: ExpressMAN

Since the last decade, interconnection of various LANs (Local Area Networks) has been an active research area. In (Borgonovo, 1987), issues of using MAN (Metropolitan Area Networks) as LAN interconnection backbone was examined and an alternative MAN architecture, ExpressMAN, is proposed as an interconnecting network. In this sub-section, the approach adopted in ExpressMAN architecture is discussed.

In the 1980s, most LANs are packet-switching systems connecting several devices located in a restricted area. The common architectural design of these systems uses the broadcast capability as the simplest way of avoiding switching centers such as those used in wide area networks (WANs). However, it is commonly understood that the broadcast capability places a severe constraint on the bandwidth utilization in the metropolitan service (Borgonovo, 1987). In fact, due to the high degree of traffic locality expected, bandwidth is better exploited by networks which allow parallel

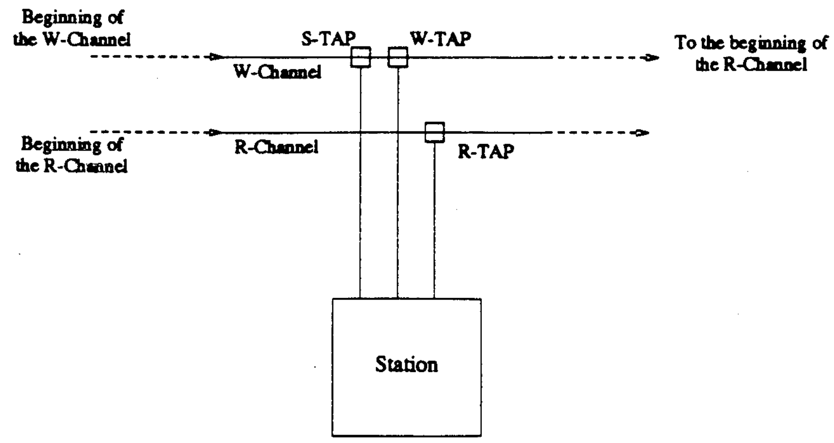

Beginning of the W-Channel
S-TAP
W-TAP
To the beginning of the R-Channel
W-Channel
Beginning of the R-Channel
R-Channel
R-TAP
Station

Figure 20: Expressnet Unidirectional Taps

transmissions (defined later in this sub-section). In general, the architectures proposed for MANs that allow parallelism basically fall into two classes:

- The first class is *Hierarchical structure* which is a collection of LANs connected by backbone networks (Yang and Mark, 1986). High speed bridges or gateways are the building blocks of these structures. No constraints are placed on the architecture of the connected LANs. Thus, it is a very general approach. However, to design the bridges and gateways fast enough to cope with the expected traffic amount is a challenging task.
- The second class envisages the MAN as a whole, rather than a collection of LANs. For example, it includes networks using mesh topology (Maxemchuk, 1985) similar to the wide-area packet switching networks; but, unlike the packet switching WANs, they do not need to store packets. These networks present regular topologies and use simple routing algorithms which allow for very fast switching. Transmission parallelism and a high degree of connection reliability are assured by multiple paths existing between any pair of nodes.

ExpressMAN is an alternative MAN architecture which lies halfway between the above two classes. While it can be seen as a collection of LANs connected by a backbone broadcast network, it uses neither bridges nor network storage facilities. Instead, the elements connecting a LAN to the backbone are simple switches which can be dynamically set to change the topology of the network which appears either as a unique bus or as several disjoined bus structures for local connection.

Specifically, the ExpressMAN access algorithm is based on the Expressnet access mechanism

(Fratta et al., 1981; Tobagi et al., 1983). Expressnet is a unidirectional bus broadcast network. Its transmission medium consists of two main channels, a Write (W) and a Read (R) channel, which are joined by a connection channel in such a way that the signal on both R and W channels always flows

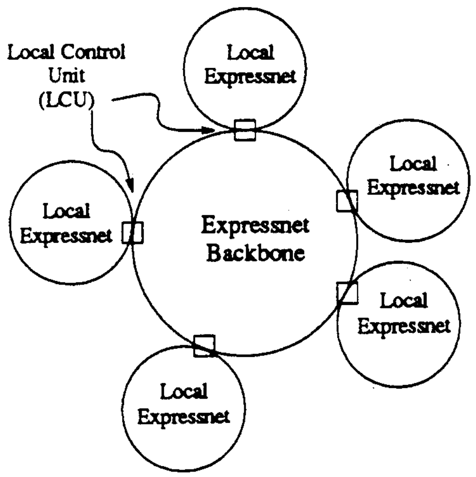


Figure 21: ExpressMAN Topology

in the same direction. Stations are connected to the bus through three unidirectional taps (Fig. 20): a Transmission tap (W-TAP) on the W channel, a Sense tap (S-TAP) to sense upstream activity on the W channel, and a Read tap (R-TAP) on the R channel. An implicit token which starts cyclically at the upstream end of the W channel, propagates downstream to collect the transmitted packets sequentially which form a packet *train*. At the end of the W channel, the train is forwarded to the R channel and received by all the stations. A station starts transmitting upon detecting EOC (end of carries on the S-TAP) from the W-channel. In the meantime, it continues to monitor the upstream activity on the S-TAP. If a BOC (begin of carrier) is detected, the transmission is aborted and the procedure is repeated. The EOT (end of train) on the R-channel event is said to occur whenever the silence period observed at the end of a packet exceeds a preset number of seconds. The EOT event triggers the transmission of stations in the same way as EOC does and a distributed train restart occurs.

ExpressMAN, which uses the aforementioned Expressnet access mechanism, distinguishes between long distance and local trains. Routing facilities are completely avoided. A simple routing task is performed by the medium access unit of the transmitting station which is only required to recognized whether the packet is either directed locally or not and to transmit it onto the correct train. A very appealing feature of the network is that local communication is allowed to take the bandwidth unused by long distance trains, thereby increasing bandwidth efficiency. This is so called *parallel transmission*.

ExpressMAN are configured in a Read and a Write channel as in Expressnet. It is based on several local structures which connect the stations belonging to the same cluster, and a backbone structure connecting the clusters as shown in Fig. 21. The connections between local and backbone buses is achieved through local control units (LCUs). The LCUs are switching units able to set different connections between the backbone channels and the local channels. When connections are set up as in Fig. 22, the network appears as a single Expressnet structure and allows long distance

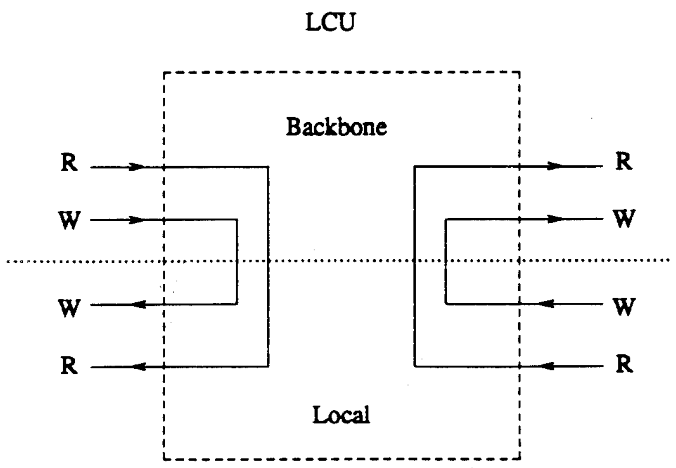


Figure 22: Long Distance LCU Configuration

(LD) trains to circulate. LD trains are those who have different source and destination clusters. In Fig. 23, the LCUs are set up to form multiple local Expressnets, one for each cluster of stations, which allow different local (LO) trains to circulate independently.

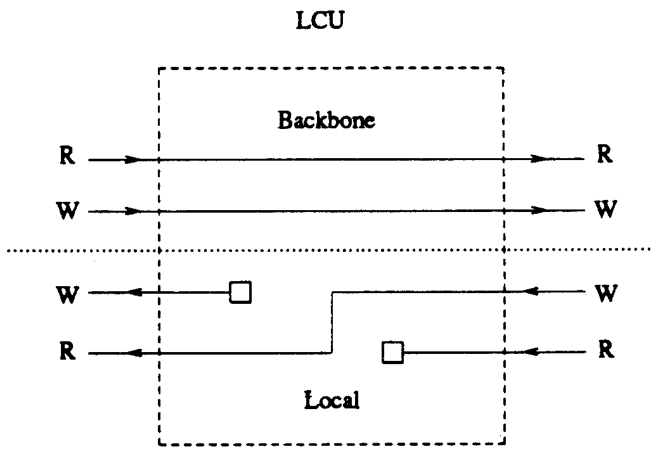


Figure 23: Local LCU Configuration

The ExpressMAN access protocol has two cycles and the network configuration is alternatively changed by LCUs at the start of each cycle. In the first cycle, an LD train is launched which propagates on the LD network configuration. In the second cycle, which lasts a fixed amount of time, one or more LO trains are launched separately at each cluster and propagate on the LO configurations, thus enforcing parallel transmissions.

The information needed to drive both the station access units and LCUs is extracted, as in Expressnet, from the carrier sense signals available on the Write and Read channels. LCUs sense on the Read channel at the downstream end of the local bus. A means to distinguish between LD and LO trains must be provided (e.g. different locomotive lengths may be used).

By means of simple switches, ExpressMAN can be dynamically rearranged either as a single bus or as a multiple bus structure to connect users grouped in clusters. It retains the features of Expressnet, but in addition, exploits the traffic locality by allowing parallel transmissions. Still, it is a broadcast network, and therefore does not require routing facilities (such as bridges). It is an alternative to a hierarchical structure which is usually used in interconnection of homogeneous LANs. Finally, it is reported in (Borgonovo, 1987) that ExpressMAN achieves a bandwidth efficiency close to that attained by a hierarchical MAN with less complexity (since there is no routing needed in ExpressMAN mechanisms).

4.7 Other Examples of LAN Interconnection

The examples described in previous sub-sections are only a few of the hundreds of previous work done by many researchers. Still, there are many other related works which cannot be discussed in this article due to its space limitation, but are important to the evolvement of the internetworking research. For example, in (Dowd and Jabbour, 1987), Dowd *et al.* exploited LAN interconnection from a different angle. Their approach is based on a hypercube topology and independent of the protocol implemented at each LAN. Their hypercube approach combines the advances in static interconnection topology, demand assignment multiple access protocols, and the availability of high bandwidth fiber optic channels, to create a structure capable of interconnecting a large number of LANs with heavy traffic at a significant cost reduction over the point-to-point interconnection.

In the rest of this sub-section, a few more of internetworking examples will be briefly described. Interested readers should refer to the original work for details.

In (Albanese et al., 1986), Albanese *et al.* proposed an architecture for a transparent gateway between a MAN composed of a Fasnet operating at 140 Mbits/s and LANs composed of Ethernets operating at 10 Mbits/s. The use of this gateway is also considered for interconnecting MANs. In (Chiou and Liu, 1986), Chiou and Liu proposed a GATENET transportation system to provide internetwork voice/data communication based upon a unique hierarchical integration of the internetwork gateways. In (Sze, 1985), Sze proposed a MAN that covers a campus environment and integrates voice, data and video under a single transmission medium. The transmission is achieved through a single media-access scheme. The use of bridges as integral components in the media-access control layer is discussed.

In (Stockman, 1993), Stockman described the versions and goals for building the Global Internet eXchange (GIX) whose concept is based on the vision of the homogeneous Internet to give maximal connectivity and maximal flexibility. Maximal connectivity is where anyone can talk to anyone anywhere at any given time. Maximal flexibility is where every organization that participates can easily set its own rules and easily implement them. In (Cain and Cerf, 1983), Cain and Cerf outlined the principles on which the U.S. DoD packet Internet architecture is based and characterized some of the protocols which implement the architecture.

In (Casoria, 1985), Casoria investigated the aspects of interconnection and services in office communication. A private network connected to public telecommunications networks is considered. Also, in (Chew, 1983), Chew reviewed the methods for network interconnection and the characteristic difference between LANs and public telecommunication networks. The key issues associated with LAN-public network internetworking are discussed. A preferred approach and its state of development were also described.

In (Adams et al., 1981), Adams *et al.* described an architecture for interconnecting a set of local area networks to cover a wide area. Links are provided by a broadcast satellite network. In (Dunn, 1984), the general approaches taken in the design of a LAN to WAN gateway are discussed and a specific implementation based on proprietary products is described.

Finally, in (Cheng and Robertazzi, 1987), Cheng and Robertazzi provided an annotated bibliography of interconnection articles. In the bibliography, the interconnection of local area networks and interconnection by means of metropolitan area networks are emphasized.

5 Interconnection of High-Speed Networks

Standards for high-speed subnetworks with efficient medium access control (MAC) protocols have been developed, including the fiber-distributed data interface (FDDI) and the distributed queue dual bus (DQDB). Prototype LANs and MANs operating at speeds in excess of 1 Gb/s are also being constructed (Kung, 1992). Such LANs can be used as backbones for the interconnection of existing LANs. LAN interconnection is normally carried out using either MAC-layer bridges or network-layer routers. In terms of high speed transmission, the routers have the drawbacks of a lower frame forwarding/relaying rate, even though they provide added functionality over bridges. Consequently, bridges have become increasingly important for the interconnection of high-speed networks in the gigabit range (Ahmad and Halsall, 1993).

On the other hand, using different technology, the ATM (asynchronous transfer mode) network has also been proposed to interconnect high speed MANs such as DQDB networks to cover larger areas. In 1988, CCITT agreed that B-ISDN would be based on ATM, basically because of the high flexibility of ATM to support all potential services (Prycker, 1990). In 1993, CCITT agreed on a series of 13 standards, defining all the basics of ATM (Prycker, 1990). In the meantime, MAN standards and SMDS/CBDS (Switched Multimegabit Data Services/Connectionless Broadband Data Service) standards were defined by IEEE and ANSI. They are aligned to ATM, like MAN, or are planned to use ATM as a bearer for SMDS/CBDS (Prycker, 1992).

In this section, high speed bridges/routers, ATM backbones and some related issues on high speed network interconnection are discussed.

5.1 Routing

Routing has been one of the most studied issues in bridged high-speed local-area networks. Among different routing schemes, source routing (Sunshine, 1977), transparent spanning tree (IEEE, 1989), and transparent source routing (Ahmad and Halsall, 1993) are discussed in this sub-section.

5.1.1 Source Routing

Source routing provides a table-free solution that applies especially well to multiply connected mesh topologies and dynamic station location (Pitt and Winkler, 1987; Sunshine, 1977). In this scheme, each frame contains a description of the route consisting of the LANs and bridges the frame must traverse. Numbering the LANs and including their numbers in the frame make loops and multiple paths possible and numbering the bridges and including their numbers make parallel bridges between the same two LANs possible. LAN numbers can be arbitrary binary numbers, but must be unique in the bridged LAN. Bridge numbers can also be unstructured and need to be unique only between the same two LANs.

A route from a source station to a destination station consists of a sequence of *route designators*, each comprising a LAN number and an adjacent-next-bridge number. A bridge scans the *routing information field* of a frame for its bridge number between the numbers of the two LANs it joins to determine how it should forward the received frame.

The routing information is acquired dynamically by the LAN stations, which in cooperation with the bridges, learn, maintain, and transmit the routing sequence. This scheme is called source routing because the source (transmitter) of every frame specifies the route to its destination. The concept applies to other types of networks, not just high-speed local-area networks (Sunshine, 1977; Haltzer et al., 1981).

In the following, the example from (Pitt and Winkler, 1987) is used to demonstrate the source routing algorithm. In Fig 24, the bridged LAN consists of five bridges and four LANs: two buses and two rings. Each LAN has a unique identifier (1 to 4) and the bridges are labeled B0 through B4. The bridges are labeled uniquely for convenience. In fact, only B0 and B1 need to be distinctly numbered. LANs 1, 2 and 3 form a loop and bridges B0 and B1 connect ring 1 and bus 2 in parallel. A route through LANs 1 and 3, abbreviated (1, 3), or via LANs (1, 2, 3) allows communication from station S1 on ring 1 to station S2 on ring 3. The LAN sequences that each bridge is part of are listed as follows:

- B0 - (1,2) (1,2,3) (1,2,3,4) (3,1,2) (4,3,1,2)
- B1 - (1,2) (1,2,3) (1,2,3,4) (3,1,2) (4,3,1,2)

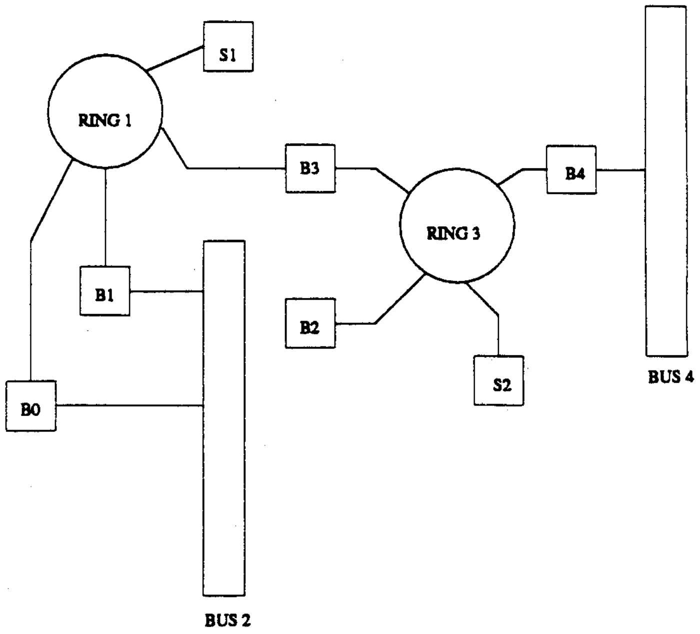


Figure 24: Source Routing Example

- B2 - (2,3) (2,3,4) (2,3,1) (1,2,3) (1,2,3,4)
- B3 - (1,3) (1,3,2) (1,3,4) (2,1,3) (2,1,3,4)
- B4 - (3,4) (1,3,4) (2,3,4) (1,2,3,4) (2,1,3,4)

A bridge does not need to store or know these sequences, only the unique portion that is common to all LAN sequences that use it. For example, bridge B2 recognizes and forwards frames whose routing information includes LAN sequences (...,2,3,...) or (...,3,2,...).

At initiation, each source end station learns the routing information by broadcasting a route discovery frame to explore all possible routes to the required destination; from the replies, the source station choose the optimal route, which is retained and used for subsequent transmissions to that destination. Source routing greatly reduces bridge processing overheads for routing frames, since bridges need not keep a routing table. However, the burden is shifted to end stations. As a result, applications need to be aware of the network interconnection status thereby increasing the complexity of application programs.

5.1.2 Transparent Spanning Tree

The transparent spanning tree algorithm involves three phases: learning end-station address, frame forwarding, and resolving possible loops in the topology by participation in the spanning tree algorithm.

On each port, a forwarding database is maintained by reading the source address from the head of all receiving frames. The bridge forwarding database contains a list of the group and individual station addresses, along with information relating these addresses to one or more of the bridge ports. When a frame is received at a port, the database is searched to determine if the destination MAC address in the frame header is present.

If the destination address is present in the database, the frame is relayed to the appropriate port. If not, the frame is forwarded on all bridge port except the one on which it was received. All the bridges in a LAN implement the spanning tree algorithm and protocol to configure the active topology into a single spanning tree; selected bridge ports are set into the blocking state so there is at most one route between any two end stations. The information relating to the derivation of the spanning tree is exchanged between bridges using special frames known as bridge protocol data units (BPDUs).

The advantage of transparent spanning tree scheme is that the routing need not be handled by each source node. Thus, it is *transparent* to end stations; i.e. end users need not be aware of the topologies of network interconnection and from users' viewpoint, every station is connected to the same single network. Consequently, the complexity of application programs is greatly reduced.

On the other hand, one serious drawback of this approach is that when the interconnected network grows, the forwarding database can become very large which will impose high processing overhead on all the bridges in the interconnected LAN. Clearly, this issue will become more important as existing LAN's are upgraded and gigabit backbones introduced. Bridges will thus become bottlenecks.

5.1.3 Transparent Source Routing

To overcome the drawbacks of pure source routing and transparent spanning tree routing algorithms, the *transparent source routing algorithm* was proposed in (Ahmad and Halsall, 1993) to combine the advantages of both of the previous routing algorithms.

In the transparent source routing algorithm, each bridge port, for forwarding and routing purpose, maintains two types of database: a local database, and a remote database. The local database only maintains station addresses that are active and attached to the corresponding local subnetwork which directly connects to the bridge through one of its ports; the remote database contains routing information only for those remote stations that are (active) communicating partners with any of its local stations. The routing information for a particular destination station consists of a series of identifiers of intermediate subnetworks and bridges. If more than one bridge is attached to the subnetwork, only the bridge providing the optimal route can route frames to a particular remote station. Hence, each entry in the remote database of a bridge port has a route status field indicating whether or not the bridge has a valid route to that station.

For a particular station with known routing information, even though there are multiple bridges to the same subnetwork, only the bridge providing the best route keeps the station address and associated routing information; it is given status V (valid) in the remote database of the port attached to that subnetwork, and the corresponding entry in all other bridges is set to D (denied). For a remote station for which a route is not yet known, only one bridge initiates route search for it and the station address in the remote database is assigned a status US (under-search) until the routing information is discovered; all other bridges on the same subnetwork keep the corresponding entry with a route status of B (blocked), and any frames addressed to these stations are discarded.

The operation of transparent source routing algorithm is as follows. When a bridge receives a frame generated by a local station, it first reads the MAC destination address in the frame header. The local database is searched to determine if the destination address is present. If it is, the destination station is attached to the same local subnetwork and the frame is discarded. If it is not, the local database of other ports are searched; if the destination address is present, the frame is forwarded to the appropriate subnetwork.

If the destination address is not present in any of its local-port databases, the remote database of the source port is searched. If a valid route (status = V) to the destination station is present, the routing information is attached to the head of the frame and it is forwarded directly by the bridge. All the intermediate bridges, after reading the routing information present in the routing header, simply relay the frame to the next subnetwork on the route. The destination bridge removes the routing information from the frame and relays it to the destination segment.

If the destination address is found in the remote database of the source port with status as US, or not found in any of the databases, the frame is broadcast to all bridge ports except the one on which it was received (assuming a loop free topology). If the destination address is found in the corresponding remote database but the route status is B or D, the frame is discarded.

With the proposed scheme, only the local bridges of the two communicating stations keep routing information; unlike transparent bridges, all other intermediate bridges along the route between the two communicating stations keep nothing, which is particularly important for bridges connected to high-speed backbones. Also, unlike source routing, a station communicates with all other stations in a transparent way, i.e. the other stations do not find routes or retain any routing information.

5.2 High Speed Backbones

A backbone is a subnetwork to which no end systems (workstations, servers etc.) are normally connected. Backbone subnetworks are used soly for internetwork traffic. For interconnecting only a small number of network segments (as in a single building), backbones of the same type as the interconnected networks (CSMA/CD, token bus or token ring) can be used. As the number of interconnected subnetworks increases, there comes a point at which the transmission bandwidth required by the backbone to meet the internetwork traffic starts to exceed that available with the basic network types. To overcome this problem, backbones based on newer high-speed network types are used. In general, such a high speed backbone subnetwork needs to interconnect subnetworks that are usually spread over a wider geographical area than a single building. An example is the FDDI LAN that interconnects various smaller LANs in a university campus or manufacturing plant. The resulting network is then known as an *establishment* or *site backbone*. Another example of the backbone network is using ATM networks to interconnect various high speed LANs across cities or states. *Information superhighway* has been proposed in recent years based upon this long-distance high-speed backbone concept.

Traditionally, TDM multiplexers, multi-protocol routers, or X.25 switches have been used as multi-protocol backbones. In (Heinanen, 1992), using frame relay as a multi-protocol backbone interface is discussed. Prior to the advances in the ATM technology, frame relay (FR) is considered in 1992 as the most versatile and most cost effective of all the available backbone technologies.

Frame Relay (FR) is a CCITT standardized connection-oriented data link access protocol (ITU, 1991) that can be thought of as a light weight version of X.25. It operates with variable length packets called frames. FR interfaces providing LAN interconnect services support a maximum frame size of at least 1600 octets. FR guarantees that frames arrive at the destination error-free and in correct order, but it does not deal with lost frames nor try to recover erroneous frames. The frame

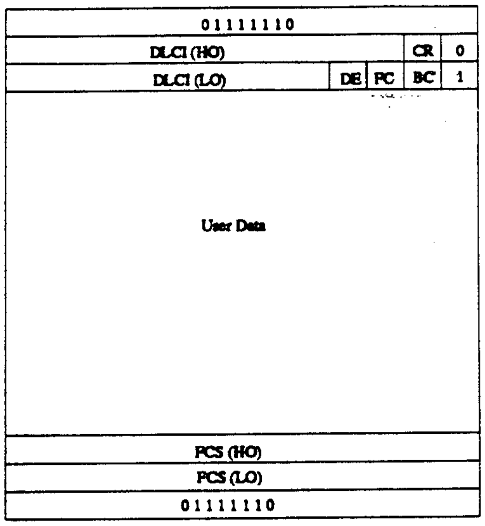


Figure 25: FR frame format

format of FR is shown in Fig. 25 where the notations used are explained as follows:

- DLCI - Data link connection identifier
- CR - Command/response indicator
- DE - Discard eligibility indicator
- FC - Forward explicit congestion notification (FECN)

- BC - Backward explicit congestion notification (BECN)
- FCS - Frame check sequence

The detailed operations of FR are discussed in (ITU, 1991).

In general, the key advantages of FR over other types of backbone is that on top of data link layers, customers can carry any network layer protocols and adopt any addressing schemes without fear of interference with other customers. Further, the selection of customer premise equipments (CPEs) is not limited to routers, since FR support is readily available, for example, for Ethernet and Token Ring bridges, X.25 switches, host computers, and communication controllers for SNA. In addition, because of FR's connection oriented nature, a per-frame accounting option is easy to implement.

Despite of the advantages of FR technologies, with the increasing number of applications, higher and higher bit rates are required at network interconnections. Technology with higher transmission rate than that of FR is desired. Thus ATM has since received more and more attention as a network interconnection backbone. ATM employs cell relay (CR) technology which offers a higher transmission speed in the ranges as high as some Gb/s. In fact, in today's interconnection of high speed LANs, most of them are based on B-ISDN techniques, using the ATM (asynchronous transfer mode) cell relay principle. There are two options when using the cell relay technology (Prycker, 1992). The first option, which is initially applicable to a metropolitan area is based on the MAN (metropolitan area network) technology as standardized by IEEE 802.6. This option is based on the DQDB (distributed queue dual bus) principle, and offers to the customer both high speed data interconnectivity and isochronous services. To cover larger areas than metropolitan ones, MANs can be connected by high speed leased lines, or by an ATM backbone network.

Alternatively, another option of using the cell relay technology, with some complementarity to MANs, is called the ATM cross-connect network (Beau, 1991), which provides high speed interconnection over a flexible virtual network. The virtual network is realized by providing a mesh network of semi-permanent *virtual paths* between end-users. As ATM inherently offers the flexibility of switching information *asynchronously* independently from the external transmission bit rate, various transmission systems, of up to 622 Mbps (Prycker, 1992) can be accommodated. A multitude of telecommunication and data services at any bit rate, fixed or variable, (voice, video, LAN and mainframe interconnection) can be transported over this network. This option of using cell relay offers the customer semi-permanent high speed connectivity. It can also be used as an enhancement to MANs, to provide a backbone to connect multiple MANs. Such an ATM cross-connect network can be enhanced with SMDS/CBDS (Switched Multimegabit Data Services/Connectionless Broadband Data Service) connectionless servers, to offer the customers full international high speed data connectivity.

Both of the aforementioned cell relay options use the ATM transfer mode of B-ISDN and thus allow a gradual evolution to a full B-ISDN, in which each service can demand dynamically its virtual connection (or multiple virtual circuits for multimedia services).

5.3 Bandwidth Allocation in ATM Backbones

In the evolutionary path to B-ISDN based on the Asynchronous Transfer Mode, the interconnection of LANs and MANs is one of the first high-speed applications. One of the difficulties in internetworking between LANs/MANs and ATM is that LANs and MANs use connectionless transfer mode, whereas ATM is a connection-oriented technique. In ATM, a virtual channel must be established before user data can be transmitted. To facilitate the transition between connectionless and connection-oriented transfer, two different solutions have been proposed in (Biocca et al., 1990; Crocetti et al., 1991; Gerla et al., 1991; Mongiovi et al., 1991; Isaku and Ishukura, 1990; Tirtaamadja and Palmer, 1990).

The first solution is to connect the internetworking unit (IWU) across the ATM network via permanent virtual paths (VP) or permanent virtual connections (VC), thus making the connectionless protocols completely invisible to and independent of B-ISDN. This approach has certain advantages for a small number of IWUs with large traffic between each other. The second solution is based on a direct support of connectionless data services by CLSFs (connectionless service functions) within B-ISDN (shown in Fig. 26) as recommended by CCITT Recommendation I.327 (Schodl et al., 1993). These CLSFs connected to each other through VCs/VPs are able to route each cell according to

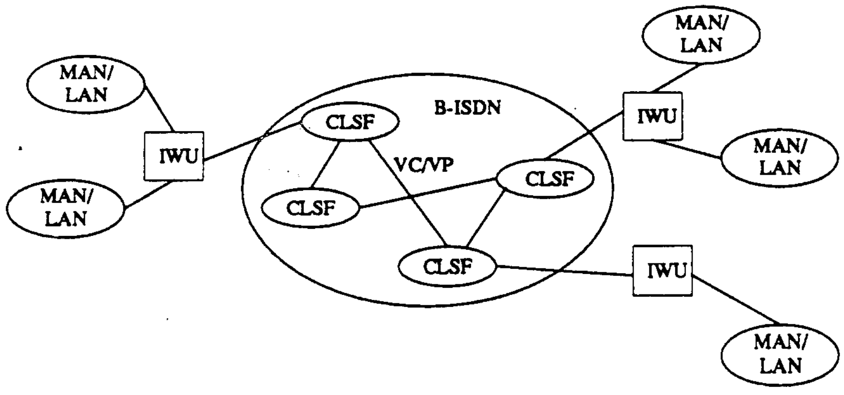


Figure 26: Direct Support of Connectionless Services

its address information. Here, it is assumed that the IWU performs traffic filtering by observing the MAC (media access control) destination address. Only those frames/cells which have a *remote* destination address are forwarded. Only the first cell of a packet is used by the CLSFs to route the packet on an appropriate VC/VP to the next CLSF. Succeeding cells with the same MID (message identification) are transmitted according to the already existing routing entry to reduce the processing overhead.

To utilize the transmission capacity efficiently, the bandwidth allocation issues in both solutions have been actively studied. In (Mongiovi et al., 1991), a bandwidth allocation mechanism for the first solution is proposed. This mechanism allows the remote IWU to size the transmission bit-rate of the VCs/VPs according to the traffic needs based on traffic measurements and IWU buffer occupancy. The main disadvantage is the huge buffer required due to the fact that a burst exceeding the VC bit-rate must be buffered in the IWU during the renegotiation phase which may take a few hundred milliseconds. This results in large queuing delays. Alternatively, in (Yin and Hluchyj, 1991) a different strategy is described. In this alternative strategy, bursts may be transferred through the ATM network without prior bit rate reservation, even if the peak rate allocated to the VP is exceeded, as long as there is enough spare bit-rate in the path. The available bit-rate is periodically distributed to all IWUs by the ATM nodes using a broadcast mechanism called *bandwidth advertising*. Thus, when a short-term overload situation occurs, the burst is not buffered in the IWU waiting for more bit-rate to be allocated; rather, it is allowed to overflow into the network. Long-term overload situation will be met by requesting more bit-rate on the VP. As an advantage to the previous scheme, cells will not suffer excessive delay in the IWUs. However, as reported in (Schodl et al., 1993), the main drawback of this scheme is the condition that low priority cells would not have any guaranteed Quality of Service (QoS).

Due to the drawbacks of bandwidth allocation schemes of the first solution, a different scheme is thus proposed in (Schodl et al., 1993) for the second solution. This bandwidth allocation scheme for direct provision of connectionless service is based on the bandwidth allocation mechanism on the VC/VP connecting the IWU and the first CLSF. Its goal is to achieve a high utilization of the transmission capacity while keeping packet lost and delay limited. Its basic idea is to size the bit-rate according to the bit-rate requirements of the traffic which is routed to the CLSF. The estimation of the bit-rate requirements is based on the arrival rate as well as the IWU buffer occupancy. The bit-rate requirement is determined as the average bit-rate of previous M (a changeable parameter) intervals. When the new estimation exceeds the previous one by more than a certain value, more bit-rate on the VC/VP to the first CLSF will be requested. On the other hand, bit-rate is deallocated only if the buffer occupancy is below a predefined threshold. During network congestion, a request for more bit-rate capacity may be denied with a certain probability.

Similar strategies may be used between the CLSFs. It is shown in (Schodl et al., 1993) that with proper choice of parameters and threshold values, the performance of this scheme can be very high. Also pointed out is that, a good compromise can be achieved using a strategy which allocates permanently a certain amount of bandwidth (which behaves ideal for short burst), and requests additional bandwidth if required (which behaves best for long burst). This strategy also reduces significantly the waiting times of packets in the IWU. Buffers must be allocated to cope with the excess information volume which arrives during the delay of an allocation request.

5.4 SMDS on Top of ATM

Switched multimegabit data service (SMDS) will be offered as a high-speed connectionless public packet-switched data service. SMDS is defined to be synergistic with the embedded base of customers' high-speed networking equipment, and has characteristics that match the services offered by LANs and MANs. In this sub-section, the design in (Crocetti et al., 1993) to implement SMDS on top of the ATM network is presented.

The main challenge of implementing SMDS on top of ATM stems from the fact that ATM uses connection-oriented technique. To provide a connectionless service on top of ATM, some additional user plane functionalities are needed to complement the ATM adaptation layers (AAL) connectionless data transfer. At a minimum, the connectionless data transfer protocol must provide for the routing and addressing (including source address validation, source/destination address screening) of connectionless, variable length packets, transferred from one source to one or more destinations without the need to establish an end-to-end connection. As pointed out in (Crocetti et al., 1993), other functions that this layer should support are: selection of QoS parameters for protocol data

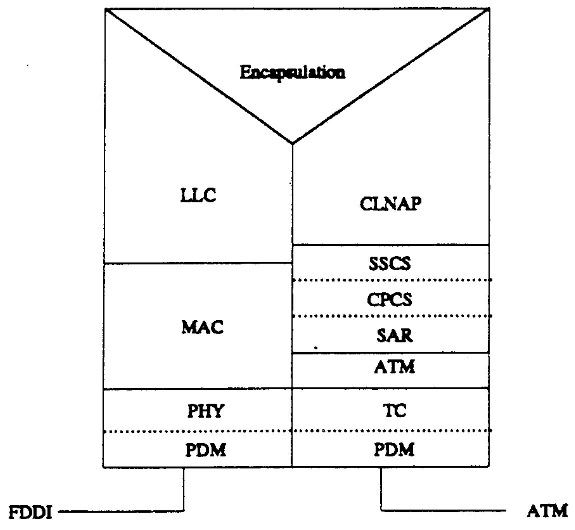


Figure 27: Bridge IWU Protocol Stack

unit (PDU) loss probability and transit delay; access class enforcement; selection of higher layer protocols; and the multicast options. These functions must be performed by the layer above the AAL, denoted as the connectionless network access protocol (CLNAP) layer. This layer definition is independent of the functions performed by the IWU. Typical protocol structures of the bridge and router that include the CLNAP layer are shown in Fig. 27 and Fig. 28, respectively.

In Fig. 27, the bridge can be used to support an *extended* LAN service across geographically distributed homogeneous LANs. A host in such LAN can address any other host, either local or remote, simply using its MAC address. The MAC address of each host belonging to the extended LAN must, of course, be unique. In general, the address transparency may be obtained either by header translation and broadcast or by encapsulation (Crocetti et al., 1993). However, the header translation and broadcast approach, used in the LAN environment is not practical in ATM because of the intrinsic inefficiency of broadcast in a mesh network. The encapsulation approach, more viable in the ATM environment, requires address mapping at the IWU: an address translation table at each IWU specifies, for each remote MAN station, the address of the IWU to which the remote MAN is connected. Encapsulation can thus be simply performed. The IWUs are connected through ATM and are identified by a MAC address. On the basis of the proper address identification, the encapsulation frame is routed to its destination. At the receiving IWU, decapsulation is performed.

To deal with a rearrangeable extendible network and with user mobility, address translation

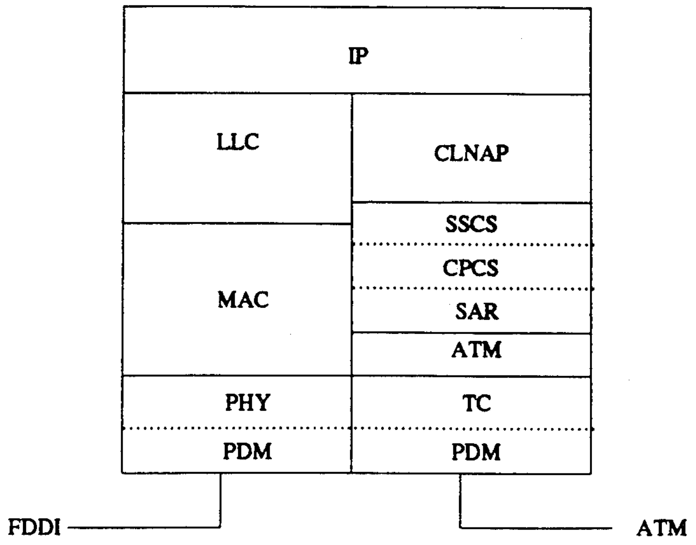


Figure 28: Router IWU Protocol Stack

tables must be updated. In principle, this can be done by observing frame header information and requiring frame broadcast when a destination host or MAN station is not listed. The response to a broadcast will allow all involved parties to update their tables.

Alternatively, routers (Fig. 28) are generally used to interconnect hosts belonging to heterogeneous LANs and MANs. A unique internet address (distinct from MAC or MAC addresses) is assigned to each host. Such an address is of the hierarchical type: net.subnet.host. In a multilevel LAN/MAN/ATM environment, the *net* specifies the MAN, and the *subnet* specifies the LAN to which the host is connected. At each IWU, a table is needed to map the logical *net* address with the MAC address. In the router implementation, the protocol stack included level 3 functions, since decapsulation and encapsulation are performed.

The IWU is the ATM customer. In SMDS topology it plays the role of the customer premises equipment (CPE). The MAC address is processed by the SMDS switching system (SS) at level 3 of the subscriber network interface (SNI). The routing of CLNAP_PDUs depends on the particular connectionless support scheme implemented in B-ISDN. If all IWU pairs are interconnected with semi-permanent VPs (virtual path), the management entity of the IWU maintains a table mapping destination MAC address with corresponding VPIs (virtual path identification).

As described in Section 5.3, the CLSFs can be considered as the nodes of a virtual connectionless overlay network. Within this network, the main function of a CLSF node is the routing of incoming datagrams, based on the MAC destination address, which is carried in the datagram (CLNAP_PDU) header. A routing table provides the next outgoing VP for each destination. In principle, many datagrams (from different sources) can be multiplexed on the same VP, i.e. the cells forming each datagrams are interleaved. The CLSF can distinguish and properly route (and, if necessary, reassemble) these cells using the multiplexing identifier (MID), which is uniquely assigned to each datagram in the VP. Clearly, the MID will change as the datagram hops from CLSF to CLSF.

Upon arrival of the first ATM cell of a specific CANAP_PDU, the CLNAP layer of the CLSF extracts the MAC address from the ATM cell and finds the next VP toward the destination. The destination can be reached directly or via one or more intermediate CLSFs. For each PDU, the CLSF stores in a table the input VPI and MID values and the corresponding output VPI and MID values.

To offer an efficient provision of the connectionless data service features such as multicasting the CLSF, encapsulates each datagram into a new CLNAP_PDU. The header and trailer of the datagram are fully contained, respectively, in the beginning of message (BOM) and in the end of message (EOM) segments.

When a datagram arrives at the first CLSF, there are two options:

- The datagram is completely *reassembled* before being forwarded. If it comes from an SNI interface, a new header and trailer are added. The next CLSF performs only a modification of the header and trailer of the datagram, without adding new ones.
- The datagram is transmitted immediately, in a *cut-through* mode. If it arrives from an SNI interface, a new BOM is added while the datagram itself becomes the first continuation of message (COM). Similarly, the received EOM becomes the last COM and a new EOM is created. Note that all the COMs belonging to the same datagram can be efficiently routed at the SAR layer using the stored information. The next CLSF replaces the received BOM with a new one.

In both options, the CLSF in the path will deliver to the destination SNI the original datagram unmodified.

An obvious problem with the reassembling option is the increased delay and the buffer required for the reassembly of several datagrams simultaneously transmitted through the CLSF. This problem is less relevant with the cut-through option. Furthermore, to cope with the possible loss of EOM cells, a time-out mechanism must be activated at each CLSF to purge obsolete input-output VPI/MID maps, and a similar mechanism should also be present in the IWU. In the CLSF overlay network, buffer congestion may occur in each CLSF. If the sustained input rate of the traffic directed to a particular outgoing VP exceeds the VP peak rate, the CLSF output buffer dedicated to the VP can experience overflow. In this case, procedures to discard a share of the traffic must be activated. In fact, the traditional and only solution in connectionless networks is to drop datagrams. In CLSF, it may be advantageous to discard cells from the same datagram rather than from randomly picked transiting datagrams. Thus, the cell discard process in the CLSF can operate according to the congestion management strategies defined in SMDS.

5.5 Local Integrated Optical Network (LION)

There is no doubt that ATM will play an important role in high speed network interconnection. However, the technologies of ATM still need time to mature. Before then, the interoperability of various high speed local networks with existing low-speed networks will become more and more critical in supporting today's fast-paced applications. In this subsection, an example of interconnecting local integrated optical networks (LION) with a traditional network, ETHERNET, is examined

LION (Luvison et al., 1987; Roffinella et al., 1987; Antnakopoulos et al., 1991) is a network intended to integrate various services of different kinds of traffic, encompassing data, voice and images, covering areas of diameter in the range of a few hundred meters up to ten kilometers. A two-level architecture has been adopted and a high-performance medium access protocol has been developed. The technique used in LION is called *hybrid-switching*, which provides both circuit and packet switching capability. Its stream traffic is supported through a transparent *bearer* service at the MAC sublayer of the ISO model for open system interconnection and the I.450, I451 ISDN protocols. The packet traffic is supported by an OSI protocol profile, as follows: the 2a OSI sublayer (MAC) is provided through a specially developed Access Protocol (AP); it is based on the hybrid switching concept due to the integration of stream and packet traffic and the expected workload; the 2b OSI sublayer follows the LLC type 1 protocol; the network layers is based on the inactive internet protocol (IP) for the LION users and the active IP for the external users; finally, the transport layer follows the transport class 4 protocol (TR4).

The impact of the internetworking problem is mirrored on the protocol profile choice since the LION network is considered as a *distributed end system* resulting in a connectionless approach up to layer 3 (network layer). In (Antnakopoulos et al., 1991), an ETHERNET to LION gateway (ELGTW) is proposed to provide access to the network layer and supports the required address transformation and relay function.

The ETHERNET to LION interconnection takes advantage of the adopted communication protocol profile, based on ISO 8473/AD1 internet protocol. Its software structure is as shown in Fig. 29, which basically consists of three parts, namely:

Figure 29: Software Structure

- A modified version of a real-time multitasking operating system.
- The scheduler, a special developed low-level software.
- The communication protocol software.

The scheduler's drivers provide the required communication primitives and data format transformation with the external world (the LION node), during initialization for identity. They also provide quality of service and various procedural functions, required by the LION global software architecture.

The ETHERNET interface is handled and managed by a specially developed kernel function, integrated in the scheduler module, in order to provide modular and efficient handling. With the specialized scheduler, the responsibilities of the kernel are:

- Initialization and testing of the ETHERNET interface.
- Local resources management, related to the ETHERNET interface.
- At the level of LLC1 sublayer, interface to the software that implements the required primitives.

The gateway follows the same protocol profile as the LION network up to the network layer. As shown in Fig. 29, the special software which provides the needed environment for the processing of the LION network packets includes AXI, GBUF and BTIM modules:

- AXI: an operating system independent module, providing the access interface between the external world and internal software modules (e.g. the IP and layer management).
- GBUF: an operating system independent module, providing the management of the buffers.
- GTIM: an operating system independent module, providing the time management functions.

The operating system (OS) communication to the system task is carried out through the environment module (ENV) in the form of primitives exchanged between each task and the OS. The responsibilities of the ENV module are:

- Interface for subsystem access by user entities and entity access by the subsystems.
- Isolation of the OS and hardware environment, providing the required format transformation, drivers, utilities and monitoring interfaces.
- Management and allocation of the system resources, especially buffers, queues and timers.
- Intercommunication between the various system tasks.
- User interface in the form of C language functions for transmitting and receiving the primitives of the various subsystems and the real-time multitasking OS.

The communication of the scheduler with the protocol processing modules is performed through messages exchanges between the tasks implementing the modules. The protocol processing modules are in a *dormant* state until they receive a message to process a packet. When they *wake up*, they perform the packet processing, send it back to the scheduler and are put again in the dormant state, awaiting a new message.

Finally in (Antnakopoulos et al., 1991), the gateway performance is studied through simulation and the selections of implementation parameters for best performance are given.

As can be seen, this gateway is constructed in ad hoc ways by detailed studying of the target network and protocols. Just like many other ad hoc conversion algorithms (Francois and Potocki, 1983; Groenback, 1986; Rodriguez, 1988; Rose and Cass, 1987), the complexity of the gateway construction is usually high and the resulting converter/gateway is expensive and limited. Consequently, a different approach, formal methods for protocol conversion, has been considered as an alternative. It has been noticed by many researchers that formal methods reduce the complexity of gateway construction significantly. In the second part of the article, various formal methods for protocol conversion are presented and discussed.

6 Formal Approaches to Protocol Conversion

Formal protocol conversion algorithms are those which take two arbitrary protocols and some conversion requirements as input to generate a converter as outout. Existing formal methods on protocol conversion can be categorized into two groups according to how the conversion requirements are specified:

1. Conversion Specification: In this category, the conversion requirements specify *directly* how the conversion should be performed in the resulting converter. The requirements are usually specified in the same level of details as the protocol specifications themselves.

2. Service Specification: In this category, the conversion requirements specify the resulting service desired from the overall protocol system with no concern as to how the conversion should be performed. The algorithms must derive the conversion information from the requirements to construct the converter.

For ease in analysis and discussion, the *communication finite state machine* (CFSM) is used as a model for specifying protocols and converters to be discussed in both categories. In the CFSM model, each *protocol entity* is represented by a finite state machine (FSM). Its transitions are triggered either by some internal events or by sending/receiving a message to/from another entity. Together, transitions from all entities of a protocol perform the task of data communication function through message passing. In the rest of this section, each category is described in a separate subsection.

6.1 Conversion Specification Approach

A conversion specification is the conversion requirement which, in this category, specifies the mapping of transitions/messages between two target protocols to achieve protocol conversion functions. Both the target protocols and conversion specification are specified in the same level of details. Since there is no formal methods to obtain the conversion specification, it is usually derived heuristically by studying the given target protocols. In general, methods in this category take a conversion specification and two target protocol specifications as input to derive the desired converter as output.

6.1.1 Okumura's Conversion Seed Method

The conversion seed method was proposed in (Okumura, 1986), which is among the first that derive protocol converters formally and becomes the basis of many other formal methods proposed thereafter.

In this method, a conversion seed X, in the form of a communication finite state machine (CFSM), is used as a conversion specification to specify the constraints on the order in which some messages in the given protocols, P and Q, can be exchanged within the converter C. The given protocol P (Q) consists of two communicating protocol entities P_a (Q_a) and P_b (Q_b) which are specified in CFSM. The method consists of three steps and two major operations $\cdot$ and $*$ which are applied to the protocol entities (CFSMs) P_b, Q_α and X. The detailed definitions of $\cdot$ and $*$ are given in (Okumura, 1986).

1. $\cdot$ is an arbitrary shuffle operation that is applied to P_b and Q_a. The result is a CFSM, R ($= P_b \cdot Q_a$), which includes all the possible combination of transitions from P_b and Q_a.
2. $*$ is an intersection operation which applies the constraints implied in the conversion seed X to R such that those transition sequences that do not follow the orders specified in X are removed. The result is another CFSM, T ($= (P_b \cdot Q_a) * X$). It is claimed that if there exists a converter for the given protocols P and Q and the conversion seed, then there exists a converter D which is a sub-CFSM of $(P_b \cdot Q_a) * X$
3. The last step of the method is to remove repeatedly deadlock states from the resulting protocol entity, $(P_b \cdot Q_a) * X$, until there is no deadlock state in it. If the result is a null CFSM, there is no converter for the given conversion seed X. Otherwise, the resulting CFSM is a converter.

As an example, consider deriving a converter C for protocols P and Q with the given conversion seed, shown in Fig. 30. Protocol P is a simple ack-nack protocol and protocol Q is a poll-end protocol. In the protocol specification, the minus (-) sign on an edge indicates the sending of a message and the plus (+) sign on an edge indicates its reception. In protocol P, every message sent is acknowledged by the receiver. If a message is garbled, the receiver can send a Nack to ask for re-transmission of the previous message. In protocol Q, once the sender (Q_a) receives the poll message from the receiver (Q_b), it keeps sending data messages until there is no more to send. Finally, an *End* message is sent

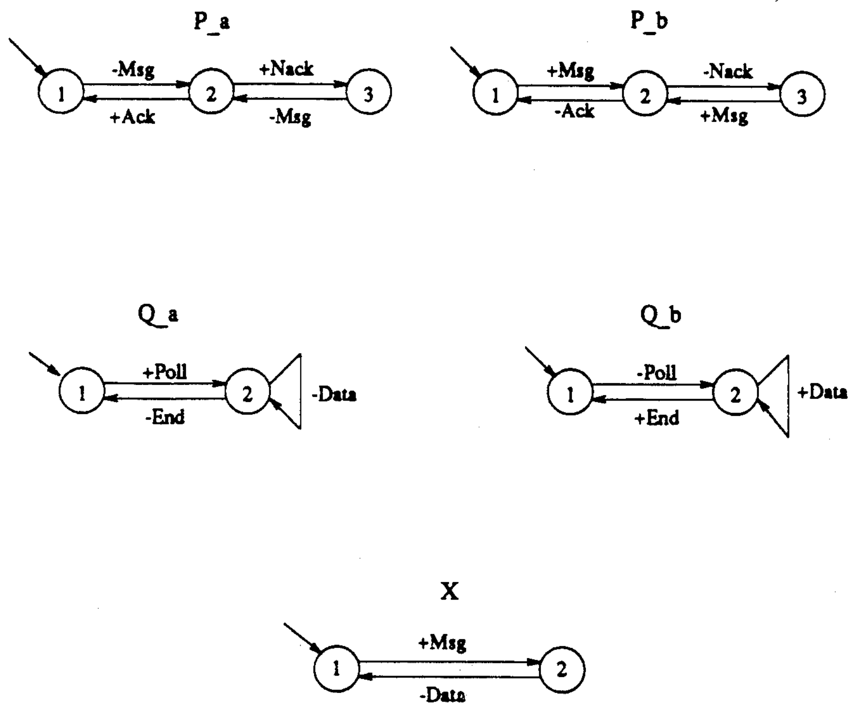


Figure 30: Example for Conversion Seed Method

to indicate the end of transmission. Note that the conversion seed X is given as an input to the algorithm, which specifies the *constraints* of the conversion: each message *Msg* received by P_b must be converted into a *Data* message to be sent by Q_a.

Apply the conversion seed algorithm to the given input, the result is shown in Fig. 31. Note that the capability of sending *Nack* message from protocol P is excluded from the converter C during the derivation. This implies that this converter does not have the error-recovery capability which is an undesirable result.

In fact, the loss of capability in this example is the result of the incompletely specified conversion seed X. Note that, only the data transmission capabilities of the target protocols is specified by the conversion seed X. Since the conversion seed X does not specify the error recovery (i.e. -*Nack*) capability, the converter derived does not include it. Selection of a conversion seed is the most important step in this method which would determines the correctness of the result. Unfortunately, there is no formal algorithm for the derivation of a *correct* conversion seed. As a result, the conversion seed is chosen heuristically by studying the target protocols, during which human errors can be easily introduced. Therefore, although the conversion seed method proposed by Okumura can be easily automated, its most serious shortcoming is the lack of a formal algorithm for the selection of a correct conversion seed.

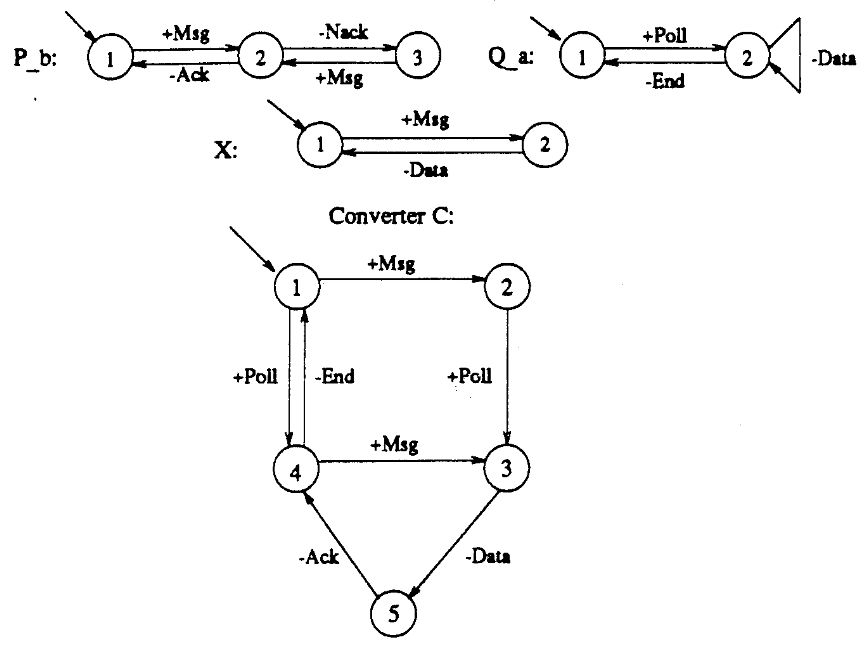


Figure 31: The Converter

6.1.2 Yao and Liu's Modular Method

The modular method proposed in (Y. W. Yao and Liu, 1990a) is to provide a systematic way to analyze the target protocols for deriving a conversion specification. In this method, the target protocols are first decomposed into distinct functional modules. Each pair of functionally compatible modules are then compared and analyzed for deriving a compatible functional converter. Since each decomposed functional module is only a part of the original protocol specification, the complexity is comparably less. Therefore, when analyzing the compatible functional modules, fewer human errors would be made during the derivation of a functional converter specification. Formally, this method consists of four major steps:

1. Decompose the two target protocols into distinct functional modules.
2. Compare and analyze pairs (one from each target protocol) of functionally compatible modules to derive the corresponding functional conversion specification.
3. Construct functional converters to translate message sequences of each pair of compatible function modules from the target protocols.
4. Merge local functions of the target protocols with all the functional converters to form a single converter.

The second and third steps of the algorithm are essentially the conversion seed operations. However, the operations are applied to a smaller-scale protocol entities which have less complexity (than the

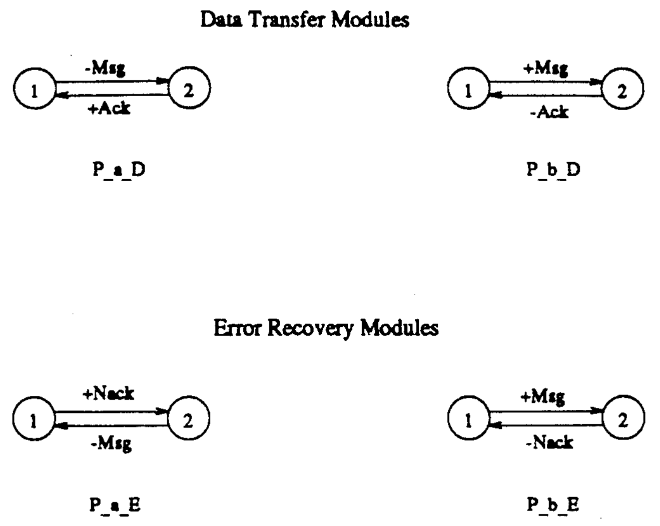


Figure 32: Decomposition of the Ack-Nack Protocol

original protocol entities). Consequently. the overall complexity is reduced. Therefore, the modular method can be considered as an improvement over the conversion seed method.

As an example, consider deriving a converter for the same example protocols of Fig. 30. The target protocol A can be decomposed into data transfer modules (P_a^D and P_b^D) and error-recovery modules (P_a^E and P_b^E) as shown in Fig. 32. On the other hand, protocol Q can be decomposed into set-up modules (Q_a^S and Q_b^S), data transfer modules (Q_a^D and Q_b^D) and release modules (Q_a^R and Q_b^R) as shown in Fig. 33. The modules P_b^D and Q_a^D are functionally compatible since they both provide data transfer capability. Using the same conversion seed X shown in Fig. 30, a converter can be derived. The final converter is shown in Fig. 34. During the derivation, it can be seen that this improvement over the original conversion seed method supports more precise specifications of the functions to be converted. Consequently, the complexity is reduced.

However, as pointed out in (Peyravian and Lea, 1993), it still requires much ingenuity in the last step of the algorithm when merging functional converters with local functions; i.e. the designer still must determine, heuristically, the execution order of all the functions (from both target protocols) to be merged, which may not be straightforward when dealing with complex protocols.

6.1.3 Lam's Projection Method

The protocol projection technique, proposed in (Lam and Shankar, 1984), was originally used to reduce the complexity of a protocol verification method. In 1986, the same technique was proposed to solve the protocol conversion problem (Lam, 1986). The basic idea of *projection* is that a property of a complex protocol can be proved by finding a *properly preserving transformation* to a simpler protocol, and by proving the property of the simpler protocol (Lam and Shankar, 1984). The simpler

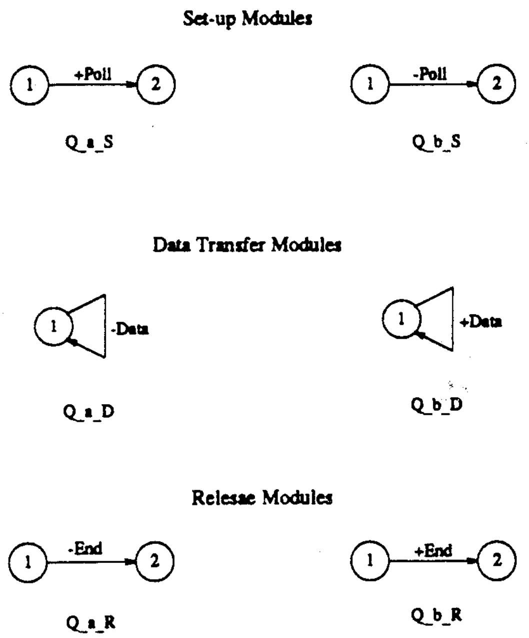


Figure 33: Decomposition of the Poll-End Protocol

protocol is called the *image protocol* of the original protocol, which has reduced *resolution*. In short, a protocol projection is accomplished by partitioning the state space of each of the protocol's CFSM. Those functional indistinguishable CFSM states in the image protocol are aggregated into the same partition, and mapped into the same CFSM state.

Thus, to solve the protocol conversion problem by the projection method, the required functionalities of the converter is given first. A heuristic search is then performed to find a common image of the given pair of protocols. If a common image entity can be found for the target protocols, an equivalence relationship between the transitions of the two protocols can be established. Consequently, this common image entity serves as a memoryless converter of the target protocols. As long as a common image entity with useful properties can be found, a protocol converter can be constructed. In fact, one can always find an image entity, say the empty entity, which is *common* to all protocols, but may not be useful, however. A candidate protocol converter is the common image protocol with the most required useful functionalities.

Finally, the missing functionalities are added to complete the converter construction since the memoryless converter constructed often may not have enough functionality for a particular application. The final converter is *correct* if it shares a useful common image with the target protocols. As a result, Lam's projection conversion algorithm requires a careful study of the nature of the protocols to be converted. In addition, the properties of the protocols need to be well understood. Therefore, a lot of human ingenuity is involved.

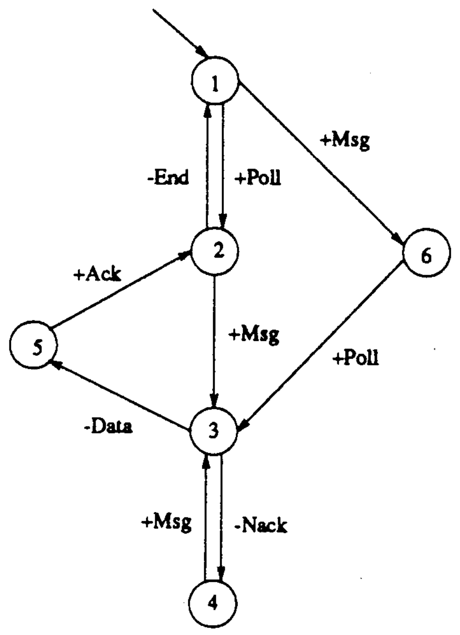


Figure 34: Final Converter Using Modular Method

6.1.4 Other Methods

Other methods in the conversion specification approach category includes the synchronization method proposed in (Shu and Liu, 1989), the parallel method proposed in (Y. W. Yao and Liu, 1990b), the transformation method in (Chang and Liu, 1990b), the complementation method in (Chang and Liu, 1990a), etc.

Again, they all suffer from the same drawbacks of the aforementioned methods: i.e. human ingenuity is needed to devise the conversion specifications, high complexity due to early involvement of implementation details in their algorithms, etc. As a result, in recent years, the trend of research in protocol conversion has shifted to a different direction, the service specification approach.

6.2 Service Specification Approach

As discussed in Section 6.1 , the main drawback of the conversion specification approach is that the implementation details are involved at the early design stages of a converter (i.e. study of the protocol specifications or implementations to obtain the conversion specification); therefore, the complexity of the algorithms becomes unmanageable when dealing with complex protocols. To reduce complexity, the service specification approach, which has received more and more attention in recent years, uses the services supported by target protocols to derive a converter. In general, a service specification method treats the implementation details as unknown *black boxes* at the early stage of design; it only depicts the final behavior of the overall protocol system as a to-be-found converter from the viewpoint of the protocol service users at the *SAP* (Service Access Point). The details will be discussed in the rest of this section.

6.2.1 Calvert and Lam's Top-Down Method

In (Calvert and Lam, 1989), a top-down method is proposed to derive protocol converters from service specifications. In this method, protocols are specified using a general state transition model similar to the CFSM model, and the requirement is given as an FSM service specification which describes the service that needs to be provided by the overall protocol system. Given two protocols P (P_a, P_b) and Q (Q_a, and Q_b) and the service specification, S, the method first determines whether there exists a converter C, such that C, P_a and Q_b interact to provide the service defined by S. If so, the method produces a specification for C. In effect, the method *divides* S by the given specifications P_a and Q_b to obtain the specification C. This method is analogous to the submodule construction problem considered in (Merlin and Bochmann, 1983), which is also used by Okumura for another service specification based protocol converter construction algorithm (to be described in the next subsection). In Calvert and Lam's method, two requirements, *safety* and *liveness*, are considered for the correctness of a converter:

- Inductively construct an entity C_0 that satisfies the safety requirement. The entity C_0 constructed has the largest trace set consistent with the safety requirement.
- Remove any state of C_0 that violates the liveness requirement of the service specification. The resulting entity is the converter specification C.

Note that, the second step of the algorithm is essentially a verification phase of the method. Since states are removed based on liveness requirements only, there is no guarantee that the required services can be supported by C. Consequently, the usage of this approach is limited.

6.2.2 Okumura's Submodule Method

In 1990, Okumura applied Merlin and Bochmann's submodule construction algorithm (Merlin and Bochmann, 1983) to derive a converter from service specifications (Okumura, 1990). Since the same submodule algorithm as in Calvert and Lam's method is used as the basis of this protocol conversion construction algorithm, the same issue of liveness (deadlock) exists. A verification phase is needed to verify the correctness of the constructed converter.

Compared with Calvert and Lam's algorithm, Okumura's method is less in computation complexity due to the fact that the service specifications are used to generated a converter in Okumura's method whereas in the Calvert and Lam's method, the actual protocol specifications are used in the derivation of the final converter. However, since the service specifications does not specify all the necessary synchronization information of the target protocols, Okumura's method may not find a valid converter in many situations. Moreover, some improvement on Okumura's method can be done.

6.2.3 Kelekar's Submodule Method

Following Okumura's submodule protocol converter method, S. Kelekar and G. Hart proposed a generalization and extension of the submodule construction method to derive protocol converters from service specifications (Kelekar and Hart, 1993).

In this method, a number of different formulations are given for the symmetric model of communications; then, the formulations are extended to a more practical model of communication, the asymmetric model. In this context, a symmetric model treats the sending and receiving of messages as an indivisible action and in an asymmetric model, the sending and receiving of messages are represented by different transitions in different protocol entities. In other words, in the symmetric model, a message can be exchanged between two communicating entities only when both entities are ready to send/receive the message, and in asymmetric model, the sender can start a send action anytime when it is ready while the receiver will then execute the corresponding receive action at a later point in time.

The safety and progress properties of the method are verified. To satisfy the safety property, a verification phase is used to remove unspecified receptions after a preliminary converter is generated. Also, to ensure infinite progress, an iterative procedure is then used to repeatedly modify the resulting converter. There is no guarantee that this iterative procedure will always terminate. However, it is pointed out that when this procedure does terminate, the resulting converter contains the progress property.

It has been shown that there exist an infinite number of possible converters in this approach for a given pair of protocols and service requirements. To obtain a minimum cost converter, where the cost is defined as the number of states and transitions in a converter, another iterative procedure is proposed to reduce the number of states and transitions in each step. However, it is mentioned in (Kelekar and Hart, 1993) that the result of the iteration may not be unique and finding the minimum cost converter is computationally hard using this approach. In fact, since finding the minimum cost converter in this generalized submodule method requires finding the regular language with a minimum state implementation between two languages, this is an NP-complete problem. Therefore, the practicality of this method is very restricted.

6.2.4 Yao and Liu's System Graph Method

Realizing the restrictions of the submodule method, Yao and Liu proposed a different approach (Yao and Liu, 1992) to derive protocol converters from service specifications. Given two target protocols

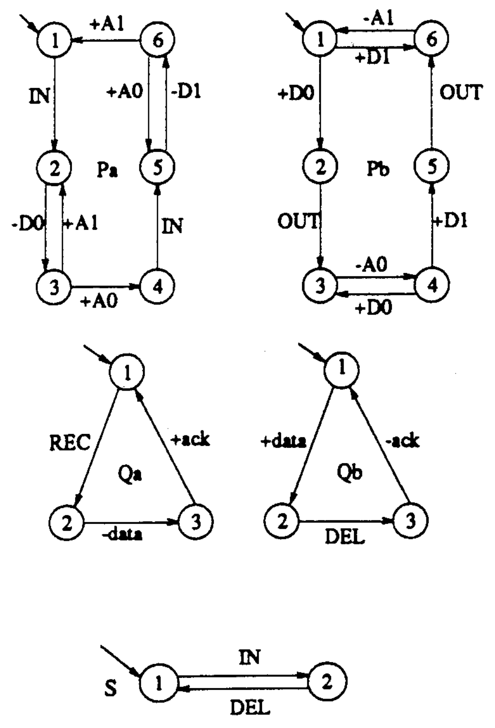


Figure 35: Example Protocol P and Q with Service Specification Requirement S

P (P_a and P_b) and Q (Q_a and Q_b), and a service specification S, the system graph method constructs a converter as follows:

1. Construct a global system graph G equal to $P_a \times S \times Q_b$
2. Transform G into a ***properly synchronized*** (Yao and Liu, 1992) system graph, G', by removing all unsynchronized states, ambiguous transitions and states that cannot reach a final state, etc.
3. Verify if the system graph G' obtained conforms to the given service specification S.
4. If G' conforms to S, derive the final converter C by projecting G' onto the transitions of P_b and Q_a.

As an example, consider the given protocols P, Q and the service specification S in Fig. 35. The converter derived using the system graph method is shown in Fig. 36.

The resulting converter is guaranteed to support exactly the service required by S. Moreover, since this method uses the service specifications instead of the protocol specifications as the given input during the derivation of the converter, its complexity is usually much less than that of Calvert and Lam's method. Moreover, because it uses only the service specifications in the derivation of

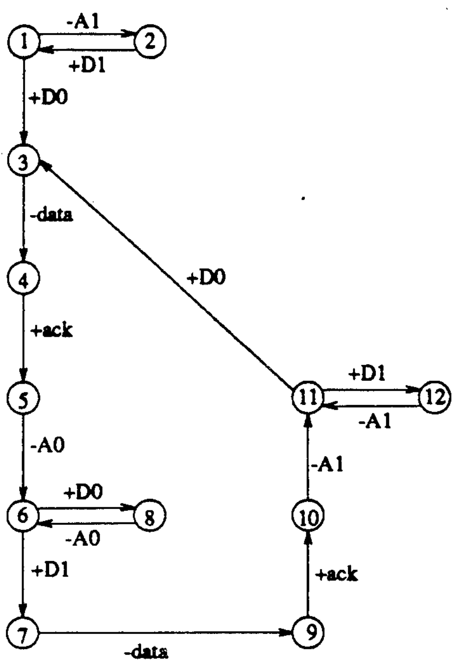


Figure 36: The Converter C

converters, it does not include in the final converter the necessary synchronization information specified in protocol specifications. As a result, the converter constructed may not be accurate and a verification of the conformance to the service specification S (step 3) is needed in the algorithm.

7 Synchronizing Transition Set Algorithm

In this section, the synchronizing transition set (STS) algorithm for protocol conversion is presented. This algorithm also belongs to the service specification approach. Unlike the four service specification methods described in Section 6.2, the STS algorithm does not have any of the shortcomings of those four methods.

The STS algorithm consists of 5 steps. It derives the conversion service specification from the given conversion service requirement; then, with the derived conversion service specification, it composes the final converter using the information from the existing protocol implementation at a later step (Step 4). The algorithm will always find a *maximum* converter that meets the requirement as specified in the conversion service requirement, and the resulting converter also always has the three most desirable properties of a correct protocol converter: conformity property (also known as maximum safety property), liveness property, and transparency property. The conformity property means that the converter so constructed supports no more and no less functions than the original protocols and conversion service requirement do. The liveness property means that the converter is free from deadlock and livelock as long as the protocols being converted and the conversion service requirement given have the liveness property by themselves. The transparency property means that the conversion algorithm does not alter the protocol entities at the end node of the composed protocol system and the conversion is transparent to the end users.

Another important capability of the STS algorithm is that it is able to handle the conversion between sequences of transitions in different protocols. This is a very important result because the semantics of a sequence of transitions/messages can be different from that of a simple concatenation of the semantics of each original individual transition/message for complex protocols, and because many of the existing conversion algorithms can only handle mapping between single transition/message. In other words, when simple one-to-one transition/message mapping between protocols is not sufficient to derive a correct converter and the conversion can only be done in a unit of sequence of transitions/messages, the STS algorithm can still derive a correct converter succeesfully. This capability of performing the conversion between sequences of transitions in different protocols is demonstrated in Example 2 in (Jeng and Liu, 1992).

Using the CFSM (communicating finite state machine) model defined in (Liu, 1989; Liu, 1990), the abstraction of the protocol specification method used in this section is shown in Fig. 37, where a protocol system consists of a set of CFSMs called *protocol entities*. Each protocol entity commu-

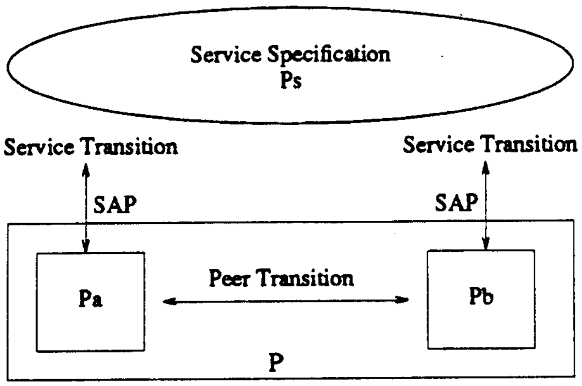


Figure 37: A Protocol System in the CFSM Model

nicates with the other through message passing. The formal definition of a protocol entity is given in (Jeng and Liu, 1992). In Fig. 37, the service specification describes how the protocol system functions from the viewpoint of the service users. It does not specify how the peer entities interact with each other.

In this section a simple example is used to demonstrate how to apply the STS algorithm to construct a converter. Before getting into the details of the algorithm, the given protocols, P and Q, and the conversion service requirement, R, are described first.

The protocol P, as shown in Fig. 38, is an ABP (Alternating Bit Protocol), in which, P_a sends data D0 and D1 alternately to P_b. "+" denotes the reception of a message or acknowledgement and

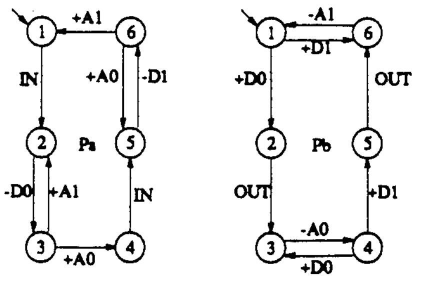


Figure 38: Example 1: The Given Protocol P

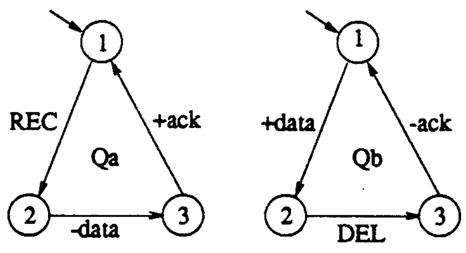


Figure 39: Example 1: The Given Protocol Q

"-" denotes the sending of a message or acknowledgement. The sets of service and peer transitions in P are given as follows:

$$\Sigma_{P_a} = \{ \text{ IN } \} \text{ (Service Transition Set of } P_a)$$
$$\Sigma_{P_b} = \{ \text{ OUT } \} \text{ (Service Transition Set of } P_b)$$
$$\lambda_{P_a} = \{ \text{ -D0, -D1, +A0, +A1 } \} \text{ (Peer Transition Set of } P_a)$$
$$\lambda_{P_b} = \{ \text{ +D0, +D1, -A0, -A1 } \} \text{ (Peer Transition set of } P_b)$$

The protocol Q, shown in Fig. 39, is a non-sequence protocol. The sets of service and peer transitions in Q are given as follows:

$$\Sigma_{Q_a} = \{ \text{ REC } \} \text{ (Service Transition Set of } Q_a)$$
$$\Sigma_{Q_b} = \{ \text{ DEL } \} \text{ (Service Transition Set of } Q_b)$$
$$\lambda_{Q_a} = \{ \text{ -data, +ack } \} \text{ (Peer Transition Set of } Q_a)$$
$$\lambda_{Q_b} = \{ \text{ +data, -ack } \} \text{ (Peer Transition set of } Q_b)$$

The service specifications, P_s and Q_s, and the conversion service requirement R are as shown in Fig. 40. The goal in this example is to derive a converter which delivers messages from P_a to Q_b. For ease in discussion, only one way of the traffic (from P_a to Q_b) is considered in this example; the traffic in the other direction (from Q_a to P_b) can be carried out in the same way.

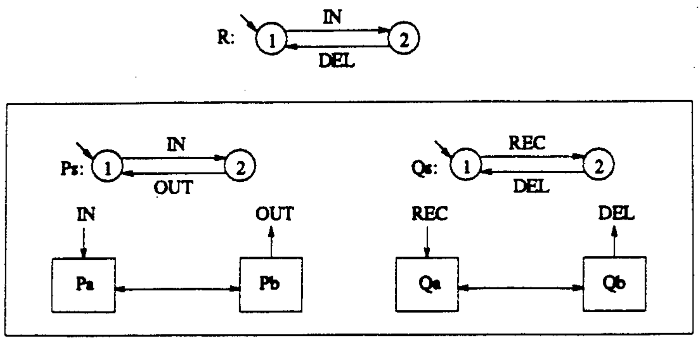


Figure 40: Example 1: The Conversion Service Requirement and Service Specifications

Note that in this example P is used in a noisy channel and Q in a reliable channel, and the *information* on the channel characteristic is not contained in the service specifications P_s and Q_s. As can be seen, P and Q result in the same service specification but different implementations. Since the goal of protocol conversion is to derive a converter for the existing implementations of P_s (ABP) and Q_s (non-sequence protocol), one will not be able to construct a *correct* converter for P and Q if no consideration is given to how P_s and Q_s are implemented. This is why it was claimed earlier that the implementation information must be included in the derivation of the protocol converter.

7.1 Step 1: Generate the Service System Graph G

In this step, the service system graph, G, is generated from the given P_s, R and Q_s by a modified composite operation, $\otimes$.

$$G = < P_s, R, Q_s > = P_s \otimes R \otimes Q_s$$

The operation $\otimes$ is formally defined as follows.

Definition 1. Modified composition operation, $\otimes$, on three CFSM entities. $G = < P_s, R, Q_s > = P_s \otimes R \otimes Q_s$ is a six-tuple $(q, \Sigma, \lambda, \delta, i, f)$, where

1. q is the set of states in G and

$$q = \{[u, v, w] |\ u, v, w \text{ are states of } P_s, R \text{ and } Q_s \text{ respectively } \}$$

2. Σ is the finite set of service transitions in G and

$$\Sigma = \Sigma_{P_s} \cup \Sigma_{Q_s}$$

3. λ is the finite set of peer transitions in G and $\lambda = \Phi$ (λ is empty)

4. δ is the transition function of G and $\delta([u, v, w], t) =$

(a) [u', v, w] if t $\in \Sigma_{P_s}$ & t $\notin \Sigma_R$ & δ_{P_s}(u, t) = u'
(b) [u, v, w'] if t $\in \Sigma_{Q_s}$ & t $\notin \Sigma_R$ & δ_{Q_s}(w, t) = w'
(c) [u', v', w] if t $\in \Sigma_{P_s}$ & t $\in \Sigma_R$ & δ_{P_s}(u, t) = u' & δ_R(v, t) = v'
(d) [u, v', w'] if t $\in \Sigma_{Q_s}$ & t $\in \Sigma_R$ & δ_R(v, t) = v' & δ_{Q_s}(w, t) = w'
(e) t is not executable at state [u, v, w] in G if none of the above four conditions is true.

5. i is the initial state of G and $i = [i_{P_s}, i_R, i_{Q_s}]$, where i_{P_s}, i_R and i_{Q_s} are the initial states of P_s, R and Q_s, respectively.
6. f is the set of the final states in G and $f = \{[f_{P_s}, f_R, f_{Q_s}]\}$ $(= \{[i_{P_s}, i_R, i_{Q_s}]\})$

The service system graph G, constructed by the modified composition operation, $\otimes$, describes how the protocols P and Q work together as a composed protocol system to perform the functions requested by the conversion service requirement R. Note that G does not contain any peer transition and it describes the behavior of the composed protocol system at the service transition level (i.e., at SAP). The result of applying the modified composition operation $\otimes$ to P_s, R and Q_s in this example

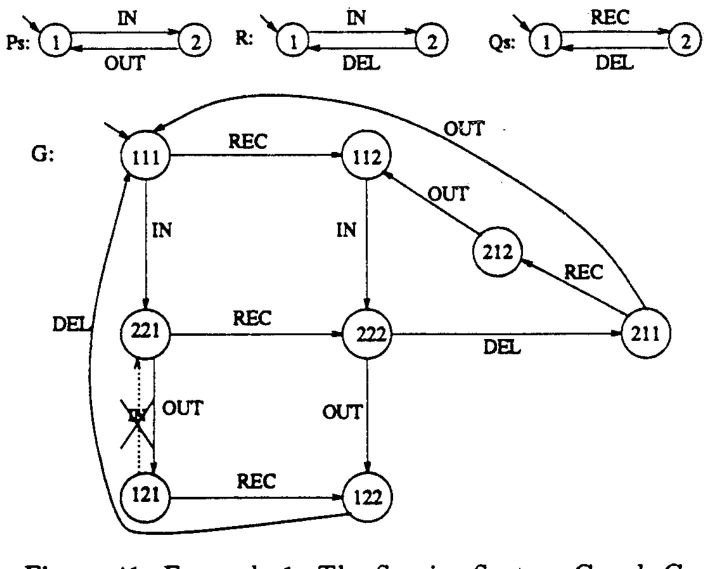


Figure 41: Example 1: The Service System Graph G

to obtain the service system graph G is shown in Fig. 41.

The difference between the modified composition operation $\otimes$ and the usual composition operation defined in (Liu, 1990) is as follows: in the modified composition operation $\otimes$, a transition t belonging to both P_s and R is executable in G only when both P_s and R are ready to execute it; i.e. both P_s and R must reach the starting state of t for t to be executable in G. The same is true for those transitions belonging to both R and Q_s. Note that this difference is enforced by items 4.c and 4.d of Definition 2.

As shown in Fig. 41, at state [1, 2, 1], transition IN to state [2, 2, 1] is not executable in G when using the modified composition operation $\otimes$, since IN $\in \Sigma_{P_s}$ & IN $\in \Sigma_R$ and IN is not executable at state 2 of R, even though IN is executable at state 1 of P_s. On the other hand, in the usual composition operation defined in (Liu, 1990), the transition IN from state [1, 2, 1] to state [2, 2, 1] would have been allowed. The transition sequence

$$p_x = \text{IN.OUT.IN.OUT.REC.DEL}$$

would become a legal path in G, if the transition IN from state [1, 2, 1] to state [2, 2, 1] were allowed in G. It is easy to verify that the transition sequence p_x does not obey the *rule* specified in the conversion service requirement, R, which essentially says that each IN must be followed by *one* DEL in a legal path. To be more specific, the projection of p_x onto Σ_R, $p_x |_{\Sigma_R}$ (= IN.IN.DEL), which has two INs and only one DEL, is not accepted by R, and as a result p_x should not be a legal path of G. Therefore, items 4.c and 4.d of Definition 2 are used to exclude from G the transition IN from state [1, 2, 1] to state [2, 2, 1], thereby making the transition sequence p_x (which does not meet the requirement R) an illegal path in G. Note that a legal path of a CFSM in this article is defined as a sequence of transitions which starts and ends at the initial state.

One may argue that G could lose some *functionalities* from the original protocols if the transition IN from state [1, 2, 1] to state [2, 2, 1] were not included. The answer to this argument is that the *capability* of executing the transition IN is not taken out from G (hence no loss of functionality) after applying the modified composition operation $\otimes$ to P_s, R and Q_s. The transition IN is indeed included at some other starting states (state [1, 1, 1] and state [1, 1, 2] but not state [1, 2, 1]) to make the final service system graph G obey all the rules specified by P_s, R and Q_s. By going through G, it is clear that the legal paths accepted by G are a mixture of the transitions in P_s and Q_s in an order that follows the sequencing prescribed by the conversion service requirement R and the given service specifications P_s and Q_s. In short, G describes how the protocols work together to provide the service required, under the constraints of the given conversion service requirement R. All transitions not explicitly specified in the conversion service requirement are by default preserved automatically. On the other hand, when it is desirable to remove some transitions from the original protocols, the conversion service requirement is used to specify explicitly such removal (as shown in Example 3 in (Jeng and Liu, 1992)).

7.2 Step 2: Derive the Conversion Service Specification M

The conversion service requirement R (hence also the derived service system graph G) only describes how the final protocol system should function and neither R nor G tells specifically how the conversion should be performed. The information on how to perform the conversion is, however, implied in G, and the goal of this step is to derive this information from G. Note that the conversion can happen only between P_b and Q_a. To know how the conversion can be done, the relationship between the transitions in P_b and Q_a must be derived first. By projecting G onto $\Sigma_{P_b} \cup \Sigma_{Q_a}$ one can derive a conversion service specification M, which has only transitions of P_b and Q_a and describes the

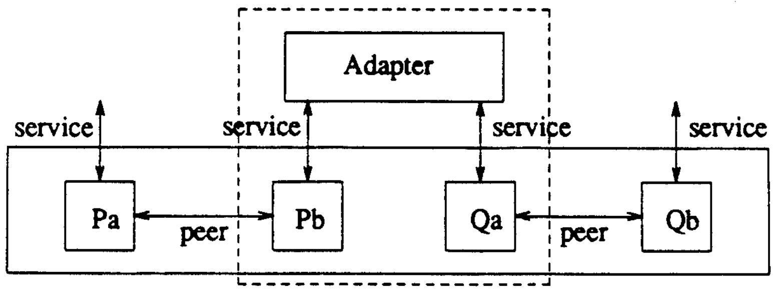


Figure 42: The Adaptor Conversion Model

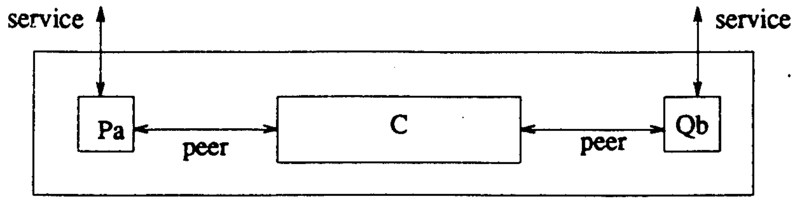


Figure 43: The Converter Conversion Model

necessary sequences of the transitions in P_b and Q_a to provide the service required in R; i.e. M $(= G\ |_{\Sigma_{P_b} \cup \Sigma_{Q_a}})$ describes how the service transitions of P_b and Q_a *interact* to perform the service requested in R. The projection operation, |, is the same as that defined in (Liu, 1990).

In fact, the adaptor conversion model is adopted in this step as shown in Fig. 42, where the

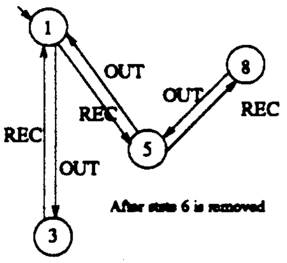


Figure 44: Example 1: The Conversion Service Specification, M

adaptor performs the conversion by synchronizing the execution of the service transitions of P_b and Q_a; i.e. the conversion service specification M specifies the proper (service) transition sequence of the adaptor for the protocol system to perform the functions requested by the requirement R. Note that M (hence also the adaptor) contains only service transitions and all service transitions are later to be removed from the final converter. In Step 5 of the algorithm when all the service transitions are removed from the converter, Fig. 42 is eventually transformed into the converter conversion model depicted in Fig. 43.

The resulting conversion service specification M is shown in Fig. 44. During the projection operation, a state with only one null outgoing transition can be removed simply by changing the ending states of all the incoming transitions into the ending state of the null outgoing transition. Similarly, a state with only one null incoming transition can be removed simply by changing the starting states of all the outgoing transitions into the starting state of the null incoming transition.

Examining Fig. 44, it is obvious that the transition sequence REC.OUT accepted by M is semantically incorrect, since it means that Q_a can send out the data to Q_b before P_b hands over the data received from P_a to Q_a. However, by examining the given service specifications P_s and Q_s and the conversion service requirement R, what they have specified are as follows:

P_s: an IN must be followed by an OUT (IN $\rightarrow$ OUT).

R: an IN must be followed by a DEL (IN $\rightarrow$ DEL).

Q_s: a REC must be followed by a DEL (REC $\rightarrow$ DEL).

Note that M is a projection of G and REC.OUT is just a sub-sequence of transitions of some legal paths of G. The transition sequence REC.OUT does not violate the rules as long as an IN was executed earlier and a DEL is executed later in the original legal paths of G (e.g. IN.REC.OUT.DEL). The given specifications do not specify specifically what the correct ordering of OUT and REC should be, and the ordering of OUT and REC is left to be determined by the implementation of the converter from the service users' viewpoint. Therefore, this further shows the validity of the claim made earlier that the information from service specification alone is not sufficient to derive the correct protocol converter and the information from the implementation must be included. Nevertheless, M tells us that the transitions OUT and REC need to be synchronized and this is what the next step of the algorithm is for.

7.3 Step 3: Generate the Synchronizing Transition Sets

The goal of this step is to derive the synchronizing transition sets from the conversion service specification M obtained in Step 2. A synchronizing transition set contains the transitions which must be executed *together* (in both P_b and Q_a) for the protocol system to correctly perform the conversion as requested by the conversion service requirement R. For ease in discussion, the transitions in a synchronizing transition set is called the *synchronizing transitions*. Essentially, the idea behind this step is to identify the aforementioned synchronizing transitions and make them executed together such that the other transitions before and after them in both protocol entities P_b and Q_a can be executed synchronously. Fig. 45, in which t_j and t_k are members of a synchronizing transition set, demonstrates how the synchronizing transitions t_j and t_k synchronize the execution of the transitions in protocol entities P_b and Q_a.

As shown in Fig. 45, when P_b and Q_a are forced to execute t_j and t_k at the same time, all the transitions t_m's that are executable only after t_j in P_b should not be executed before t_k in Q_a is executed. For the same reason, the execution of the transitions t_n's in Q_a (which are executable only

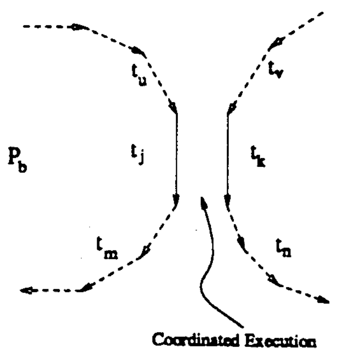


Figure 45: Concurrent Execution with Coordination of t_j and t_k

after t_k) is synchronized with that of t_j in P_b. Consequently, the execution of the transitions t_u's (t_v's) is synchronized with that of t_n's (t_m's) which belong to a different entity. Note that t_m, t_n, t_u and t_v are as shown in Fig. 45.

The generation procedure of the synchronizing transition sets is described in details in (Jeng and Liu, 1992; Jeng, 1995). Apply this synchronizing transition set generation procedure to M; the only synchronizing transition set S_1 (in this example) can be identified as follows:

$$\{ \text{REC}, \text{OUT} \}$$

One may argue that it is so trivial that REC and OUT should be synchronized and the synchronizing transition set generation procedure is redundant. This may be true for this particular example, in which there is only one service transition in each of P_b and Q_a (hence only 2 different transitions in M); therefore, there is at most one synchronizing transition set, and these 2 transitions are both in the synchronizing transition set. However, in situations in which there are more service transitions in M (for complex protocols), finding the synchronizing transition set is not always trivial, and a formal procedure to generate the synchronizing transition sets is needed. An example to illustrate this situation is given in (Jeng and Liu, 1992; Jeng, 1995).

7.4 Step 4: Converter Composition

In this step the coordinated composition operation, $\odot$, is used to derive the converter, using the synchronizing transition sets obtained in Step 3 as the synchronizing mechanism. This is also the step in which information from the protocol implementation is involved in the construction of the converter. The idea of this step is to let protocol entities P_b and Q_a execute concurrently and coordinatedly. As depicted in Fig. 45, when the coordinated composition operation, $\odot$, is applied to the protocol specifications of P_b and Q_a, protocol entities P_b and Q_a are forced to execute the transitions in synchronizing transition sets together and the coordination/synchronization between transitions in P_b and Q_a is achieved.

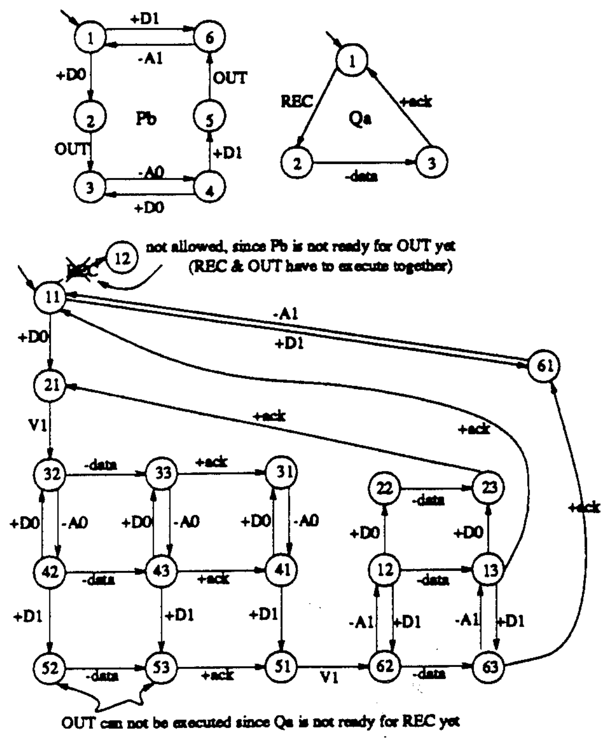


Figure 46: Example 1: The Converter C

In this example, there is only one synchronizing transition set S_1. For ease in discussion, a new transition V_1 for S_1 is created, such that the execution of V_1 in both P_b and Q_a means the execution of all the transitions in S_1 (i.e. REC and OUT) at the same time. That is, V_1 is to be incorporated into the converter by the $\odot$ operation. The new transition V_1 is called the ***virtual atomic transition***, and the following can be established:

virtual atomic transition V_1 with
$S_1 = \{$ REC, OUT $\}$
and
converter $C = P_b \odot Q_a$

The operation $\odot$ is defined as follows.

Definition 2. Coordinated Composition Operation $\odot$.
The converter $C = P_b \odot Q_a$, with the virtual atomic transition V_i and the corresponding synchronizing transition set S_i obtained in Step 3, where i = 1, 2, ..., is a six-tuple $(q, \Sigma, \lambda, \delta, i, f)$, where

1. q is the set of states in C and

$$q = \{[\mathrm{u}, \mathrm{v}] \mid \text{u, v are states of } P_b \text{ and } Q_a \text{ respectively } \}$$

2. Σ is the finite set of service transitions in C and

$$\Sigma = \Sigma_{P_b} \cup \Sigma_{Q_a} \cup \{ \text{ all } V_i\text{'s } \}$$

(The service transition set in C will become empty when all the service and virtual atomic transitions are removed from the final converter at the last step of the algorithm in Section 7.5)

3. λ is the finite set of peer transitions and

$$\lambda = \lambda_{P_b} \cup \lambda_{Q_a}$$

4. δ is the transition function of C and $\delta([\mathrm{u}, \mathrm{v}], \mathrm{t}) =$
   (a) [u', v] if $\mathrm{t} \in \lambda_{P_b}$ & $\delta_{P_b}(\mathrm{u}, \mathrm{t}) = \mathrm{u'}$
   (b) [u, v'] if $\mathrm{t} \in \lambda_{Q_a}$ & $\delta_{Q_a}(\mathrm{v}, \mathrm{t}) = \mathrm{v'}$
   (c) [u', v] if $\mathrm{t} \in \Sigma_{P_b}$ & $\mathrm{t} \notin$ any of the S_i's & $\delta_{P_b}(\mathrm{u}, \mathrm{t}) = \mathrm{u'}$
   (d) [u, v'] if $\mathrm{t} \in \Sigma_{Q_a}$ & $\mathrm{t} \notin$ any of the S_i's & $\delta_{Q_a}(\mathrm{v}, \mathrm{t}) = \mathrm{v'}$
   (e) [u', v'] if t is a virtual atomic transition, V_i,
   & $\delta_{P_b}(\mathrm{u}, \mathrm{t'}) = \mathrm{u'}$, where $\mathrm{t'} \in \Sigma_{P_b}$ and $\mathrm{t'} \in S_i$
   & $\delta_{Q_a}(\mathrm{v}, \mathrm{t''}) = \mathrm{v'}$, where $\mathrm{t''} \in \Sigma_{Q_a}$ and $\mathrm{t''} \in S_i$.
   (In this example, t' = OUT and t" = REC)
   (f) t is not executable at state [u, v] in C if none of the above is true.

5. i is the initial state of C and $i = [i_{P_b}, i_{Q_a}]$, where i_{P_b} and i_{Q_a} are the initial states of P_b and Q_a, respectively.

6. f is the set of the final states in C and $f = \{[f_{P_b}, f_{Q_a}]\}$ $(= \{[i_{P_b}, i_{Q_a}]\})$

The result of applying the $\odot$ operation to P_b and Q_a to obtain the Converter C is shown in Fig. 46.

The $\odot$ operation allows all the transitions in P_b and Q_a to execute concurrently in the sequence as the original protocols prescribed, as long as the transitions are not synchronizing transitions (i.e. transitions which do not belong to a synchronizing transition set). The synchronizing transitions in one entity have their chance of execution when the peer entity is ready to execute the corresponding synchronizing transitions in the same synchronizing transition set. In other words, one entity must wait for its peer to execute together the synchronizing transitions in a synchronizing transition set. Note that a synchronizing transition set always contains transitions from both peer entities. As shown in Fig. 46, at state [1, 1], Q_a must wait for P_b to get to state 2 before executing REC (V_1); i.e. at state [2, 1] of C, P_b (in state 2) is ready to execute OUT and Q_a (in state 1) is ready to execute REC, and both OUT and REC belong to the synchronizing transition set S_1. The waiting, however, does not change the ordering of the transition executions as prescribed in the original P_b and Q_a. In addition, none of the transitions is thrown away by the $\odot$ operation. As a result, the properties and functionalities of the original protocols are preserved.

7.5 Step 5: Remove the Remaining Service Transitions

This is the final step to derive the protocol converter. The converter obtained in Step 4 contains both service and virtual atomic transitions. Recall that there are no service users on top of the converter; therefore, all the service transitions must be changed to null transitions and then removed from the final converter using the algorithm described in (Hopcroft and Ullman, 1979). Moreover, in this example, the virtual atomic transition V_1 must also be removed since the execution of V_1 means the

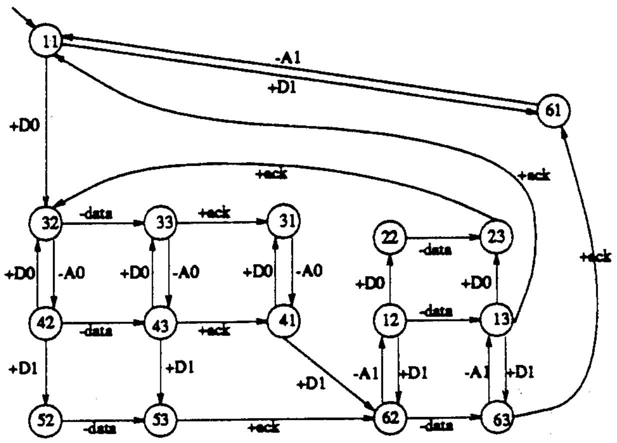


Figure 47: Example 1: The Final Converter

execution of service transitions only, i.e. OUT and REC. The resulting final converter specification is shown in Fig. 47. Note that the same technique described in Section 7.2 can be used to remove the null transitions (which are service transitions and virtual atomic transitions in this step).

To conclude this example, notice that the behavior of neither P_b nor Q_a has been changed. Instead, the execution order of the transitions between P_b and Q_a is forced in accordance with the conversion service requirement, R (hence also M). All transition sequences accepted by P_b (Q_a) are also accepted by the converter and vice versa (in terms of the projection of the legal paths of the final converter specification) while the service transitions are considered null transitions in the converter. Therefore, the properties and functionalities of the original protocols are preserved (when the conversion service requirement does not explicitly specify the removal of them). Also, the three properties for correctness are satisfied with no need for an additional validation phase. The proof of the correctness of the STS algorithm can be found in (Jeng and Liu, 1992; Jeng, 1995).

8 Protocol Conversion in EFSM Models

An EFSM (Extended Finite State Machine) is a generalization of an FSM (Finite State Machine). The major different between CFSM and EFSM is the inclusion of state variables of various types in EFSM. The values of these state variables are changed by the occurrence of events/transitions. An event can occur only if certain enabling conditions are satisfied. An enabling condition is a predicate on the state variables and is represented within a pair of square brackets in this article.

The EFSM model has more express power than the CFSM model does, and is more practical in specifying complex protocols. In this section, it is shown that the STS algorithm can also work effectively in the EFSM model. Consequently, the scope of coverage of the algorithm, in terms of number of protocols it is applicable to, is also extended.

The reason that the STS algorithm can work effectively in the EFSM model is that in the first 3 steps of the STS algorithm, only the service specifications and the conversion service requirement (in the same form as the service specifications) are used to derive the conversion service specification. The implementation details (i.e. the protocol specifications) are not involved until the last two steps when the final converter is to be derived. It is this strategy that allows the STS algorithm to work smoothly and effectively while the underlying model is extended from CFSM to EFSM. Since the extension from the CFSM model to the EFSM model affects only the protocol specification and since the high level service specification remains the same (i.e. the service provided by the protocols remains the same in both models), the first three steps of the STS algorithm need not be changed. Finally, as in the CFSM model, a converter constructed by the STS algorithm in the EFSM model also always has the three most desirable properties of a *correct* converter; namely, the conformity, liveness, and transparency properties.

The formal definitions of an EFSM protocol entity and an EFSM service specification can be found in (Jeng and Liu, 1993). In this section a simple example is used only to show how the STS (synchronizing transition set) algorithm with minor modification, can solve the protocol converter problem in the EFSM model. Since the service specification is the same in both EFSM and CFSM models and only the protocol entity specifications (implementations) differ, the first three steps of the algorithm, in which only the information from the service specification is used, remain the same. For completeness, all five steps of the algorithm are applied to the example in this sub-section to show the result.

8.1 Example in EFSM Model

In the EFSM model, a predicate is enclosed in a pair of square brackets. In both predicate and action parts of a transition, "!" and "?" are used to denote sending of a message or acknowledgement and reception of a message or acknowledgement, respectively.

The protocol P, as shown in Fig. 48, is an ABP (Alternating Bit Protocol), in which, P_a sends messages D(0) and D(1) alternately to P_b. For each message sent by P_a, an acknowledgement, A(0) for D(0) and A(1) for D(1) respectively, is sent by P_b. Variable u is used for the value of the message number sent/received and variable v is for the value of the acknowledgement number. Both u and v alternate between 0 and 1. The service transitions are IN and OUT; the service user above P_a executes IN to request sending of data and when data arrives at the other end, P_b executes OUT to pass the received data to the user above P_b.

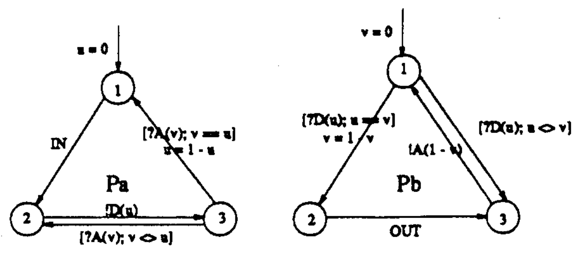


Figure 48: Example 2: The Given Protocol P (ABP) in the EFSM Model

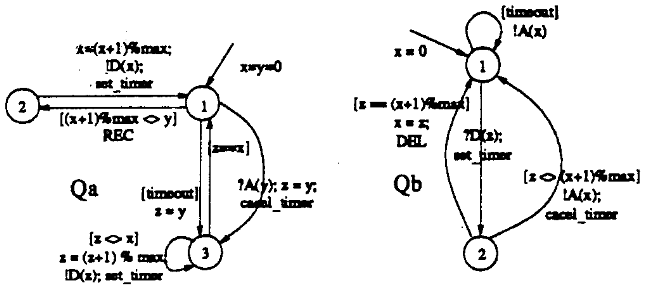


Figure 49: Example 2: The Given Protocol Q (Go-Back-N Protocol) in the EFSM Model

In P_a, when the acknowledgement number received does not match the message number sent previously (state 3 → state 2), P_a will retransmit the message. On the other hand, P_b checks the message number on each reception. If the message number received is not what has been expected, P_b informs P_a by sending the acknowledgement number of the previously received message. The ABP is used in a noisy channel where the message can be garbled but never be lost.

The protocol Q, as shown in Fig. 49, is a simplified go-back-n protocol. Note that this protocol is used in a noisy channel where data can be either garbled or lost. In the protocol entity specification, the constant "max" is the buffer size and Q_a can send up to (max - 1) messages before it starts to wait for the acknowledgement from Q_b. While max is a design parameter, (max - 1) is the n in the go-back-n protocol. In Q_a, variable x is the message number sent the last time; variable y is the acknowledgement number received the last time and z is a temporary variable. In Q_b, variable x represents the message number received the last time, and z records the current message number received. Note that "(x+1)% max" is the next message number expected.

On the sending side, the user above Q_a executes REC to request sending of data; Q_a checks if there are already (max-1) outstanding messages waiting to be acknowledged (as expressed in the predicate of the transition from state 1 to state 2) before accepting the request. Once the request is accepted the message is assigned a message number, "(x+1)% max", and sent to Q_b. Meanwhile, a timer is set for retransmission if the acknowledgement is not received within a preset timeout interval; i.e. when the timer goes off (state 1 → state 3), Q_a will retransmit all outstanding messages. Note that "set_timer" command will cancel all previously set timers. On the other hand, if an acknowledgement is received before the timer goes off, the acknowledgement number is checked and those messages

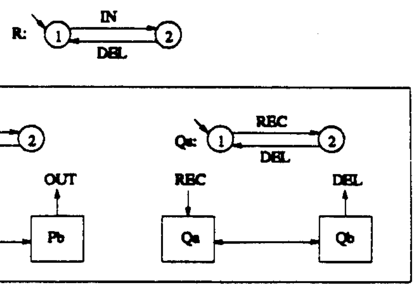


Figure 50: Example 2: The Conversion Service Requirement and Service Specifications

not acknowledged is retransmitted. An acknowledgement with a *higher* number can acknowledge all the outstanding messages with *lower* numbers. Note that the message numbers are cycled around the constant max; i.e. once a message number reaches max, it is reset to 0.

On the receiving side, Q_b simply throws away the garbled message if its message number does not match the expected number. In this case, an acknowledgement of the previously received message is sent to Q_a to trigger the retransmission. Otherwise, the service transition, DEL, is executed to deliver the received message to the service user above Q_b. Note that when the messages are received in the correct sequence, Q_b can receive at most *max - 1* messages before it sends the acknowledgement to Q_a.

The service specifications, P_s and Q_s, and the conversion service requirement R are as shown in Fig. 50. Note that the service specification in the EFSM model is the same as that in the CFSM model shown in Fig. 40 (Section 7). The goal in this example is to derive a converter which delivers messages from P_a to Q_b.

Since the first 3 steps of the algorithm remain the same as in the CFSM model, the same results can be obtained as before. Specifically, the same service system graph G (as in Fig. 41), conversion service specification M (shown in Fig. 44) and the synchronizing transition set S_1, { REC, OUT } can be obtained using the first 3 steps of the STS algorithm.

8.2 Step 4: Compose the Converter

In this step, a different coordinated composition operation $\odot$ (defined later) than that in the CFSM model is used to derive the converter from the protocol implementation/specifications and the synchronizing transition sets obtained in the previous step. Recall that in Section 7.4, it has been shown how the $\odot$ operator can be applied to protocols in the CFSM model. Due to the presence of predicates in the EFSM model, the *concatenation* of two EFSM transitions and two EFSM transition sets need to be defined before the coordinated composition operation can be applied to P_b and Q_a in the EFSM model. They are defined in Definitions 3 and 4, respectively, as follows.

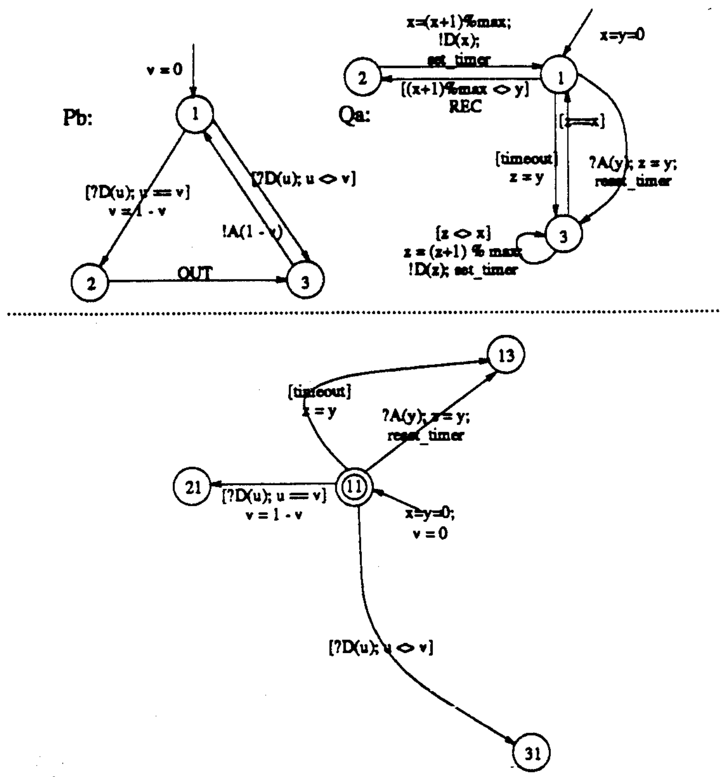


Figure 51: Example 2: The Converter C - Expansion of State (1, 1)

Definition 3. *Concatenation Operation* $\vee$.
Given two EFSM transitions t_1 and t_2, the concatenation $t_1 \vee t_2$ is defined as concatenating the predicate of one transition with that of the other and the action of one transition with that of the other. i.e.

$$\begin{gathered} \text{Given } t_1 = [Predicate_1]\ Action_1 \\ \text{and } t_2 = [Predicate_2]\ Action_2 \\ \text{then} \\ t_1 \vee t_2 = [Predicate_1,\ Predicate_2]\ (Action_1,\ Action_2) \end{gathered}$$

From Definition 3, the concatenation of two transition sets is defined as follows.

Definition 4. *Transition Set Concatenation* $\uplus$.
Given two EFSM transition sets Σ_1 and Σ_2, the concatenation set $\Sigma_\uplus = \Sigma_1 \uplus \Sigma_2$ is defined as:

$$\Sigma_\uplus = \{t \mid t = t_1 \vee t_2,\ t_1 \in \Sigma_1 \text{ and } t_2 \in \Sigma_2\}$$

Note that $\Sigma_1 \subset \Sigma_\uplus$ and $\Sigma_2 \subset \Sigma_\uplus$, since either t_1 or t_2 or both can be null transitions.

Next, the coordinated composition operation, $\odot$, is to be defined to allow protocol entities P_b and Q_a to execute concurrently and coordinatedly. The synchronizing transition set obtained in Step 3 is used as a synchronization mechanism to synchronize the order of the execution of the transitions from P_b and Q_a. Intuitively, as depicted in Fig. 45, the coordinated composition operation must force the protocol entity P_b and Q_a to execute the transitions in the synchronizing transition sets together to achieve the coordination/synchronization between transitions of P_b and Q_a. To be more specific, the transitions in each of the synchronizing transition sets are concatenated first by using the concatenation operation defined in Definition 3 (one concatenated transition for each synchronizing transition set). Then, the coordinated composition operator will compose a converter from the specifications of P_b and Q_a (and the concatenated transitions) in a way that only the *atomic* executions of the concatenated transition are allowed in the converter.

In this example, there is only one synchronizing transition set: S_1. For ease in discussion, a new transition V_1 is created to denote the concatenation of the transitions in S_1, such that the execution of V_1 in both P_b and Q_a means executing all the transitions in S_1 (i.e. REC and OUT) at the same time. Also, V_1 is to be incorporated into the converter by the $\odot$ operation. Note that V_1 is a virtual transition in the sense that it will be removed at the last step (step 5) of the algorithm. For the sake of discussion, V_1 is called the ***virtual atomic transition*** in the rest of the section. Formally, V_1 (of the example) is described as follows:

V_1 is a virtual atomic transition for
S_1 = {[(x+1)%max <> y]REC, OUT }
and
V_1 = [(x+1)%max <> y]REC $\vee$ OUT = [(x+1)%max <> y](REC, OUT)

The converter $C = P_b \odot Q_a$ and the operation $\odot$ is defined as in (Jeng and Liu, 1993). Fig. 51 shows the application of the coordinated composition operation after state [1, 1] is expanded. For easy reference, protocol specifications P_b and Q_a are shown again. Note that the transition [(x+1)%max <> y]REC of Q_a from state [1, 1] to state [1, 2] is not included since it belongs to a synchronizing transition set. Fig. 52 shows the application of the coordinated composition operation after state [2, 1] is expanded. Note that the virtual atomic transition V_1 is now included since at state [2, 1] both P_b and Q_a are ready to execute the transitions in the synchronizing transition set.

The $\odot$ operation allows all the transitions of P_b and Q_a to execute concurrently in the sequence as the original protocols prescribed, as long as the transitions are not synchronizing transitions (i.e. transitions which do not belong to a synchronizing transition set). The synchronizing transitions of one entity have their chance of execution when the peer entity is ready to execute the corresponding synchronizing transitions in the same synchronizing transition set. In other words, one entity must wait for its peer to execute together the synchronizing transitions in a synchronizing transition set. Note that a synchronizing transition set always contains transitions from both peer entities. As shown in Fig. 51 and Fig. 52, at state [1, 1], Q_a must *wait* for P_b to get to state 2 before executing [(x+1)%max <> y]REC (V_1); i.e. at state [2, 1] in Fig. 52, P_b (in state 2) is ready to execute OUT and Q_a (in state 1) is ready to execute [(x+1)%max <> y]REC, and both OUT and [(x+1)%max <> y]REC belong to the synchronizing transition set S_1. The waiting, however,

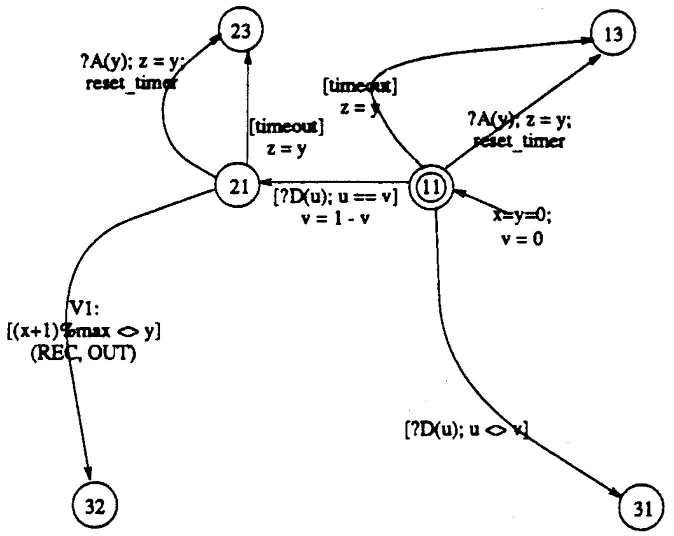


Figure 52: Example 2: The Converter C - Expansion of State (2, 1)

does not change the ordering of the transition executions prescribed in the original P_b and Q_a. In addition, none of the transitions is thrown away by the $\odot$ operation. As the result, the properties and functionalities of the original protocols are preserved. The converter after the expansion of all the states is shown in Fig. 53

8.3 Step 5: Remove the Remaining Service Transitions

The converter obtained in Step 4 contains both service and virtual atomic transitions. Recall that there is no service users on top of the converter; therefore, all the service transitions can be changed to null transitions and then removed from the final converter using the algorithm described in (Hopcroft and Ullman, 1979). In this example, the action part of the virtual atomic transition V_1, [(x+1)%max <> y](REC, OUT), can be changed to a null action since the execution of the action part of V_1 means the execution of service transitions only, i.e. OUT and REC. However, the predicate part of V_1 cannot be removed. The resulting final converter specification is shown in Fig. 54.

As a conclusion of this example, it can be seen that the behavior of either P_b or Q_a is not changed. Instead, the execution order of the transitions between P_b and Q_a are enforced as specified in the conversion service specification M (hence also the conversion service requirement R). All transition sequences accepted by P_b (Q_a) are also accepted by the converter and vice versa (in terms of the projection of the legal paths of the final converter specification) while the service transitions are considered null transitions in the converter. Therefore, the properties and functionalities of the original protocols are preserved and the three properties for correctness are satisfied with no need for an extra validation phase. The proof of correctness of this algorithm in the EFSM model can be found in (Jeng, 1995),

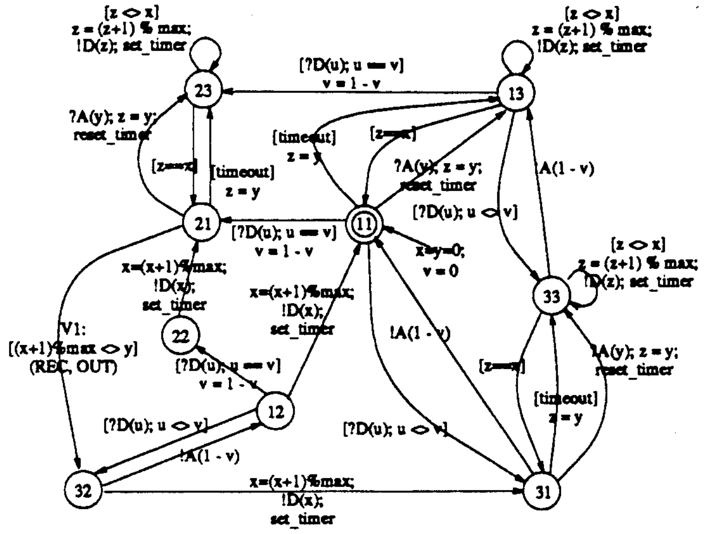


Figure 53: Example 2: The Converter - After All States are Expanded

9 Protocol Conversion in Multimedia Networks

In recent years, rapid advance in data transmission speeds is facilitating the emergence of multimedia networks. As high speed networks mature, multimedia networks are certain to become the next generation of communication networks. Out of recently proposed high speed networks such as DQDB, FDDI, ATM, etc., various issues have been examined to determine the most suitable type of backbone network for multimedia networks (Dupuy et al., 1992). At this point in time, evaluations are still inconclusive, although ATM networks have drawn the most attention lately.

In fact, different multimedia applications have different requirements with respect to transmission speed, reliability, synchronization, failure recovery, etc. The type of multimedia applications to be supported within a given multimedia network will determine the most effective communication backbone. Since there is no single high-speed network which is clearly superior to all others in all aspects, in the near future, different high-speed networks using different protocols will co-exist in an integrated multimedia network, to simultaneously support various multimedia applications.

Like the traditional heterogeneous data networks, the inter-operability among the interconnected heterogeneous high-speed networks will be the key to success for integrated multimedia networks. On the other hand, unlike the traditional heterogeneous data networks, a simulation study presented in (Jeng and Liu, 1995) shows that, due to the special characteristics (e.g., high speed and high connectivity) of these integrated multimedia networks, the traditional protocol conversion model will introduce long transmission delays and render a network system useless. This simulation study makes it clear that a different and more efficient approach is needed for protocol conversion in multimedia networks. Therefore, to further demonstrate the flexibility of the STS algorithm described in Sections 7 and 8, another modification is presented in this section to generate protocol convert-

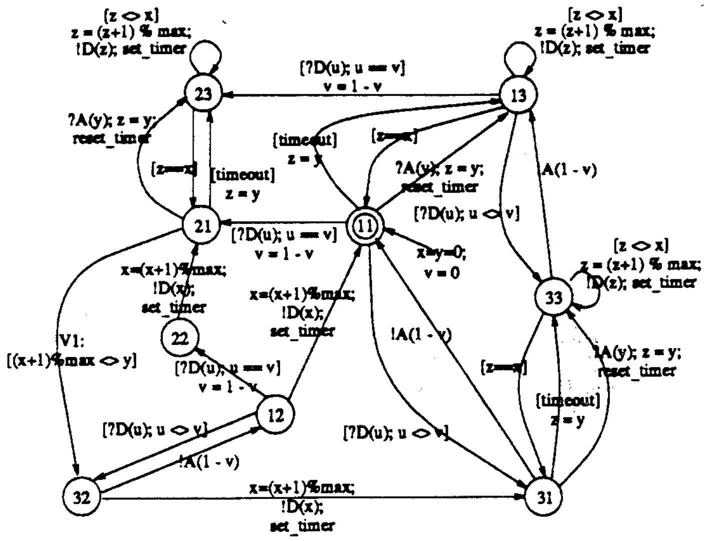


Figure 54: Example 2: The Final Converter

ers in multimedia network environments. It will be shown that with a minor modification, the STS algorithm can accommodate effectively the high speed and high connectivity requirements of a multimedia network.

In this section, the *protocol compensation* model is used instead of the traditional intermediate converter model. All the desirable features of the STS algorithm are retained: less complexity with the service transition approach; no need for an extra validation phase at the end; and the capability to convert transition sequences as a unit, etc. Finally, with an enhanced null transition removal algorithm (Jeng and Liu, 1995), which can also be applied to protocols in CFSM and EFSM models, the overall complexity of the modified STS protocol conversion algorithm becomes polynomial.

9.1 Conversion in Multimedia Network Environments

As mentioned above, rapid advance in network technologies and transmission infrastructure, using optical fiber, has resulted in an unprecedented telecommunication exploration of integrated multimedia networks. In this new generation of networks, it is the high-speed of the fiber-optic media that offers the possibility of integrating voice, graphic and video data into a single multimedia network. Technologies in many computer science areas are being explored to support the ever growing demands of multimedia applications (Rangan et al., 1993; Turner and Peterson, 1992; Hoepner, 1992; Gemmell and Christodoulakis, 1992; Moeller et al., 1990; Ramanathm and Rangan, 1993). Although the development of multimedia networks is still in its infancy, it is clear that a single multimedia network could be used to connect every HDTV, video/audio phone, and work-station in a wide area. Upon successful development of the multimedia network, each household of the covered area will be able to simply plug their HDTV sets into the network to receive TV signals. Similarly, after plugging in a video phone, video phone owners will be able to reach others just by dialing the destination numbers. In this scenario, the number of nodes connected in the multimedia networks is at least the number of households in the covered area. As a result, the connectivity (number of nodes) of a multimedia network must be much higher than a traditional data network. Therefore, in addition to high speed, high connectivity becomes another dominant characteristic of a multimedia network. Without high speed transmission, a multimedia network will not be able to support multimedia applications; without high connectivity, the coverage and usefulness of a multimedia network will be limited. The feasibility of high connectivity stems from high speed, since high connectivity without high speed transmission would result in the constant saturation of communication links within the network.

So far, a variety of high speed networks, such as ATM, FDDI, and DQDB, which exhibit speeds in the range of hundreds of Mbits/s and even Gbits/s, have been proposed as the potential backbone networks for supporting multimedia applications. There is no single network from the aforementioned list which is superior to all the others in all aspects. For example, as pointed out in (Dupuy et al., 1992), to achieve better resource utilization, the bandwidth allocation at a multiplexing point in an ATM network is dynamic. This in turn facilitates support of bursty traffic types of multimedia applications. On the other hand, an FDDI network could be favored for non-bursty transmission of voice and video packets due to its guarantees of bounded access delay. Yet under light traffic load, a DQDB network is favored for its low access delays to asynchronous traffic. Moreover, the FDDI network, which is based on token ring topology, is best suited for local area networks, while the DQDB network, which has bus topology, has been adopted by the IEEE 802.6 Working Group as a standard for metropolitan area networks. On the other hand, ATM networks are best suited for high-speed wide area networks. Because different high speed networks pose different limitations on the range of coverage, and exhibit different characteristics (e.g., access delay, bandwidth allocation), in

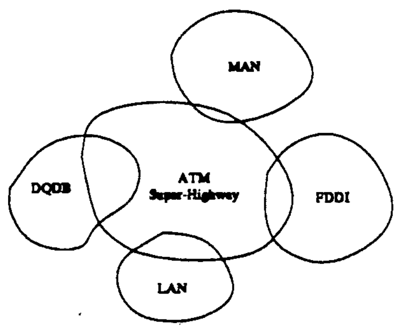


Figure 55: A Multimedia Network Model

the foreseeable future, diverse multimedia applications will require co-existence of various high speed networks in an integrated multimedia network. In fact, in a recently proposed multimedia network architecture shown in Fig. 55, various high speed networks already co-exist: an ATM network is used as a *data communication super-highway* to interconnect many LANs and MANs, which in turn may use FDDI, DQDB, and other protocols.

Nevertheless, there are mismatches among these *co-existing* high speed protocols in a multimedia network. To achieve interoperability, many researchers have started to study aspects of interconnection of heterogeneous high speed networks. However, all studies to date covering multimedia network protocol conversion (Crocetti et al., 1993; Schodl et al., 1993; Clapp, 1991; Landegem and Peschi, 1991; Mongiovi et al., 1991; Gerla et al., 1991; Biocca et al., 1990) are targeted toward some specific network systems. As a result, the proposed solutions are all ad hoc methods which cannot be applied to high-speed network systems other than their original targets. Since these protocol conversion methods do not use formal protocol conversion techniques, they are as expensive and limited as ad hoc protocol conversion algorithms in traditional data networks. In other words, to achieve interoperability among heterogeneous high-speed networks, the protocol conversion problem must be solved; and to solve it cost-effectively in multimedia networks, formal protocol conversion techniques must be used. In this section a formal approach, which is a modified version of the Synchronizing Transition Set (STS) algorithm is used for protocol conversion in the high speed/connectivity multimedia network environment.

Because of the high speed and connectivity requirements of a multimedia network, protocol conversion in multimedia networks is a more challenging problem than in traditional data networks; for the same reason, the traditional protocol conversion model shown in Fig. 56 is no longer feasible. In the traditional model, the converter/gateway is a single point of connection for the two networks with protocol mismatches. All traffic between nodes of two different networks must travel through the gateway. Consequently, the gateway becomes a traffic bottleneck. In traditional data networks, since

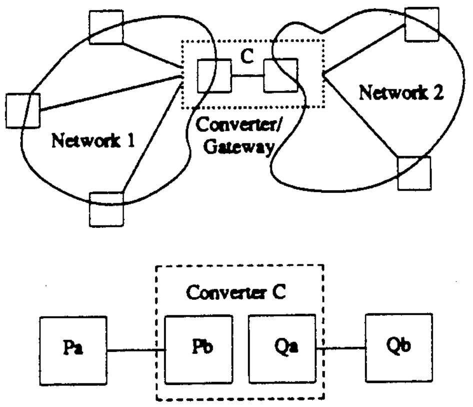


Figure 56: Traditional Conversion Model

the number of internal nodes is less than a few hundred and internetwork communication is relatively less frequent, the bottleneck effect is tolerable. In fact, the throughput requirement of traditional internetwork communication is in the range of hundreds of Kbits/s or less; therefore, in terms of network interconnection, a high-speed bridge would suffice. In the case of multimedia networks, however, high connectivity means increased probability of traffic through the converter/gateway. Coupled with the high bandwidth requirements on each media stream, the transmission delay at a single point of traffic congestion will become unmanageable. For example, in a regular DQDB network, an intermediate station which attempts to transmit a message into a slot can simultaneously read from and write into the busy bit of the slot passing by to minimize the latency incurred (Sharon and Segall, 1993). To support the required transmission speed of a multimedia network, the delays in the intermediate nodes/stations are, in general, less than a few *bits* in most of the proposed high-speed networks. If an intermediate converter node is used as in the traditional model, the delay could be as high as a few *messages*, thereby reducing the total throughput significantly. To alleviate internetwork traffic congestion, more gateways could be installed between each pair of networks; however, these gateways would still be points of traffic congestion and delay, thereby reducing the QoS (Quality of Service). Moreover, from a fault tolerance point of view, a single point of connection means that a single point of failure may cause total isolation of a subnetwork, which is very undesirable. Therefore, it is believed that when there is a choice, using a converter gateway as an intermediate node between two different networks should be avoided in a multimedia network environment. It is shown in (Jeng and Liu, 1995) that even with high throughput processors, the number of multimedia sessions a converter can support is still very limited in typical multimedia applications. Most importantly, the simulation results in (Jeng and Liu, 1995) make it clear that a more efficient approach is needed for protocol conversion in multimedia networks. Thus, in contrast to the traditional conversion model used in previous sections, a different model that performs conversion at end nodes is used in this section. In the protocol conversion literature, this category of conversion is known as the *protocol compensation* approach.

In the *end node* model, it is assumed that a node can be physically connected to a network by simply *tapping* itself into an access point of the network, in the same manner that plugging in a telephone establishes a connection. This model, without an intermediate converter gateway, performs protocol conversion at the end nodes to reduce intermediate node delay in internetwork communication.

Note that a converter gateway still must be used when the networks to be integrated are geographically separated far apart, and tapping into other networks is not feasible. However, it should also be noted that the networks in a multimedia network environment are often overlapped, much as telephone networks and cable TV networks: an average household often contains both cable TV and telephone outlets. Two networks are considered to be overlapping if they cover approximately the same geographical area and an access point of one network is always close to an access point of the other, such that a node in one network can connect to the other network without unreasonable difficulty. In this sense, if a node needs an extraordinarily long distance link (e.g., from the East Coast to the West Coast of the United States) for tapping into the other network, the tapping is considered unreasonable. The technique discussed in this section can be optimally applied when the networks to be integrated overlap.

9.2 New Conversion Model

The new conversion model used in this section is a variation of the traditional model. As discussed previously, the traditional intermediate converter in a multimedia network is not acceptable due to a bottleneck effect. To address this problem, the new model eliminates the intermediate conversion node and by contrast, conversion is performed at either of the end nodes as shown in Fig. 57 and Fig. 58. Note that it is assumed that the networks with protocol mismatches are geographically close so that a node in one network can *tap* itself into the other network to initiate internetwork

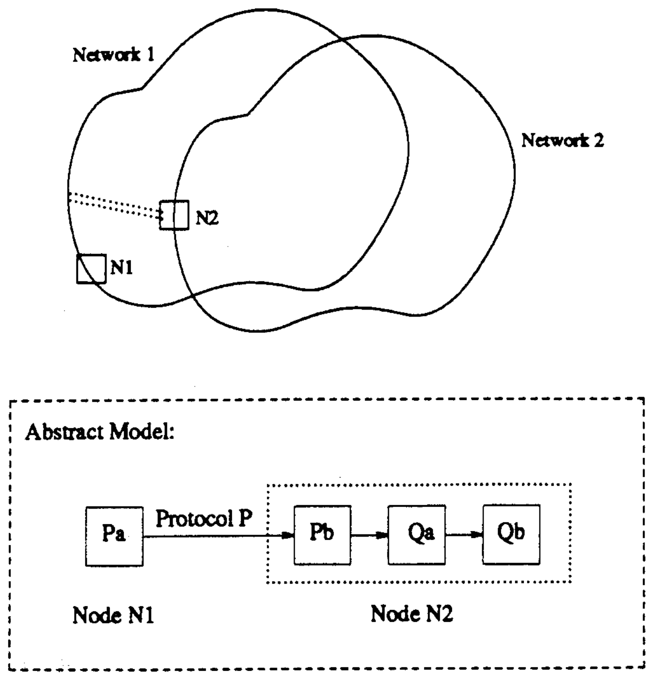


Figure 57: Option 1 - Conversion at Receiving Node

communication.

In Fig. 57, node N2 in network 2 taps itself into network 1 to initiate internetwork communication with node N1 in network 1. The dotted lines between node N2 and network 1 denote the attachment of N2 to network 1. From node N1's viewpoint, node N2 is only a newly added node in network 1 running the same protocol. N1 and N2 can be considered as two nodes connected by the same high speed links of network 1. Therefore, N1 would use the regular protocol of network 1 to communicate with N2 without knowing that N2 is a node in network 2. To keep the conversion transparent to high level application programs, node N2 performs the conversion within its low level protocol entities. The abstract model is shown in the dashed box of Fig. 57. Note that in this abstract model, protocol P (of network 1) is used between nodes N1 and N2. The transparency property requirement defined in (Jeng and Liu, 1992) is relaxed in the sense that the conversion is not transparent to the low level protocol entities of the receiving node. Still, the conversion is transparent to node N1 and the high level application programs of node N2.

In the same manner, when N1 in network 1 is the node which initiates the internetwork communication, it can also attach itself to network 2 as shown in Fig. 58. In this situation, node N1 becomes an extension of network 2 and from node N2's viewpoint, node N1 is only a new node added to network 2. The links between nodes N1 and N2 are the same as those in network 2; therefore, protocol Q (of network 2) is used instead. Likewise, the conversion is performed within the low level protocol entities of node N1. N2 does not need to be aware of that N1 is a node from another network. As a result, the protocol conversion is transparent to N2.

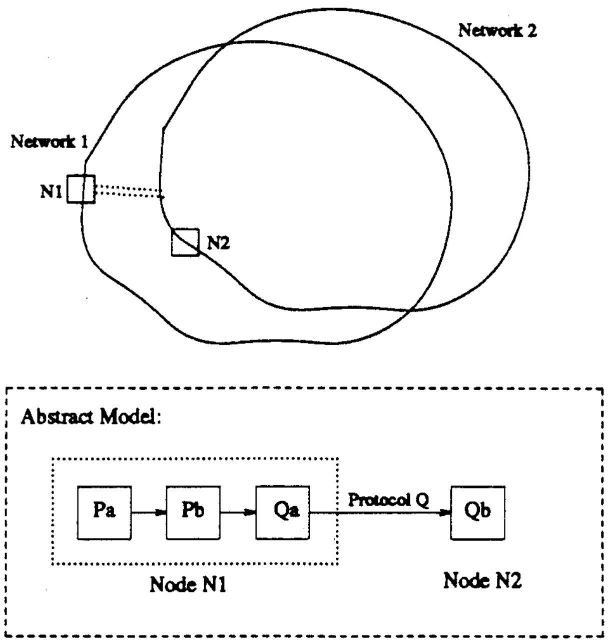


Figure 58: Option 2 - Conversion at Sending Node

The *end node* model is different from the traditional model in that there is no intermediate converter gateway between nodes of different networks. Nodes initiating internetwork communication perform the protocol conversion. Therefore, the conversion function is decentralized. Since this model requires conversion in one of the end nodes, in protocol conversion terminology, it is known as *protocol compensation*. In this decentralized conversion scheme without a single point of traffic congestion, bottlenecks are eliminated. High speed internetwork communication thus becomes more plausible. Moreover, removing the possible single point of failure that leads to nodal isolation makes this model more fault tolerant.

Despite the aforementioned advantages, there is at least one trade-off: the protocol conversion transparency requirement must be relaxed. It is up to network designers to weigh the benefits of traditional protocol conversion approach against that of the compensation model when making a choice. In the case of a multimedia network, the protocol compensation approach is viewed as a more attractive solution, because of the unacceptable intermediate converter node delay which exists in the traditional conversion model.

Both options 1 and 2 of the model work effectively in a multimedia network. The remainder of this section focuses on option 1. Readers interested in option 2 should be able to derive a similar result using the same approach as is presented in the next few sections.

9.3 Service Abstraction

In Fig. 59, a service user layer is added to the abstract models of option 1. Node N2 supports the

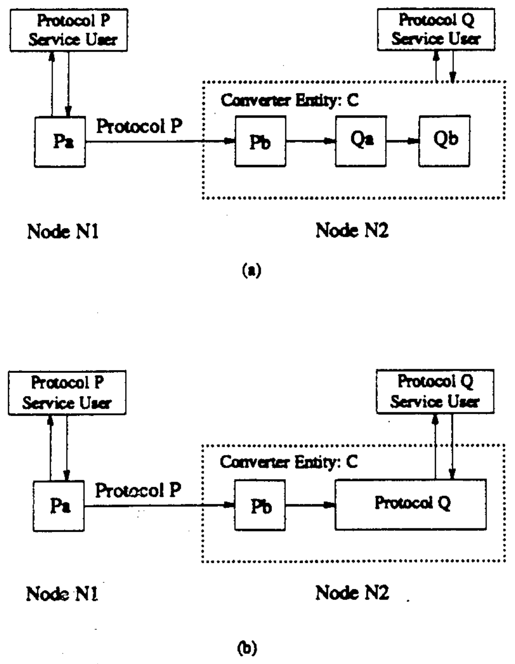


Figure 59: Abstract Model of Option 1

service interface of protocol Q, while at the same time using protocol P to communicate with its peer entity P_a. Conversion is performed by the converter entity C, which resides below the service level and is transparent to its service user (high level application), since the same service interface that does not support conversion is supported by C. The STS conversion algorithm is modified to derive the aforementioned converter entity C.

In the model in part (a) of Fig. 59, C is a composition of protocol entities P_b, Q_a and Q_b, which are the *givens* in the protocol conversion problem. Intuitively, the peer transitions between Q_a and Q_b are internal to protocol Q, which is in turn internal to converter entity C. Note that in the STS algorithm, P_b and Q_a act like service users for each other, and the conversion is done at the protocol boundary (between P_b and Q_a) through coordination of service transitions from both protocols. In addition, peer transitions are used to support the peer interface (communication links) between communicating entities of the same protocol. In the new model, Q_a and Q_b are combined inside of C such that the peer interface/communication link between them no longer exists. Thus, the interactions of peer transitions between Q_a and Q_b do not affect the conversion and can be ignored during the derivation of C. Due to the fact that the peer transitions of protocol Q are not observable from entity C's standpoint, in a later step of the algorithm (step 4), part (b) of Fig. 59 is the actual model used.

In the refined model in part (b) of Fig. 59, entity P_b and protocol Q are combined into an indivisible entity to support the peer transition part of protocol P and the service transition part of protocol Q. Moreover, inside entity C, protocol Q is considered as an entity consisting of only service transitions which interact with entity P_b to support the service interface of protocol Q. Therefore, in step 4 of the modified algorithm, it is the service specification Q_s that participates in the final composition of converter entity C. In the rest of this section the aforementioned models are used in presenting the modified algorithm.

9.4 Modified STS Algorithm

For ease in presentation, let us use Example 3 shown in Fig. 60 to give a step by step demonstration of how the STS algorithm can be modified to derive converter entity C in the new model. The given conditions of the new model are the same as before, and are shown in Fig. 60 and Fig. 61.

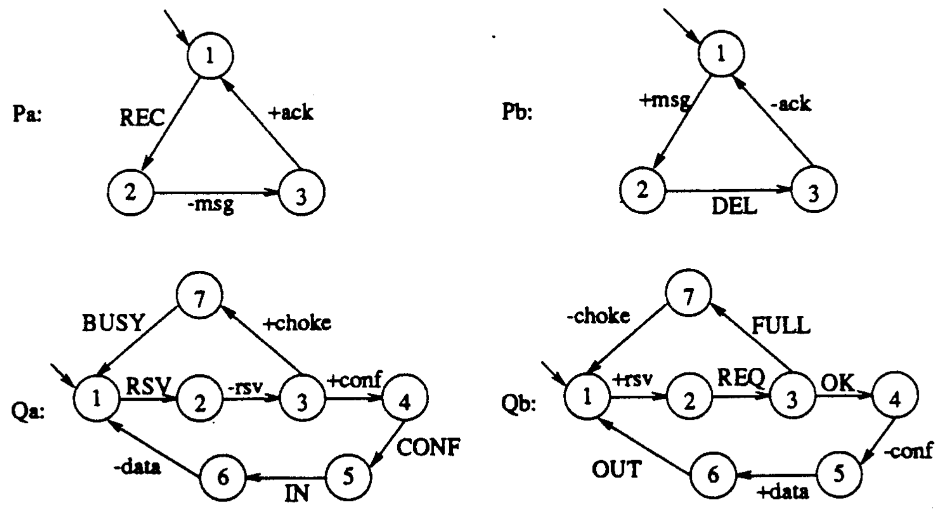


Figure 60: Example 3: Given Protocols P and Q

In the STS algorithm, the first 3 steps of the algorithm are used to derive the inter-relationship of the service transitions at the protocol boundary, between the service transitions of P_b and Q_a. In the new model, the conversion within C is also done at the boundary in a similar sense, between P_b and protocol Q, via the coordination of service transitions from both protocols. Therefore, the first 3 steps remain the same. The result of applying ⊗ (defined in Section 7.1) to P_s, R and Q_s to obtain the service system graph G is shown in Fig. 62.

The initial projection of G onto the service transitions of P_b and Q_a is shown in Fig. 63, in which states [1, 1, 2] and [2, 3, 3] have only one outgoing and one incoming null transition, respectively; therefore, they can be removed along with the incidental null transitions as described in Section 7. The result is shown in Fig. 64.

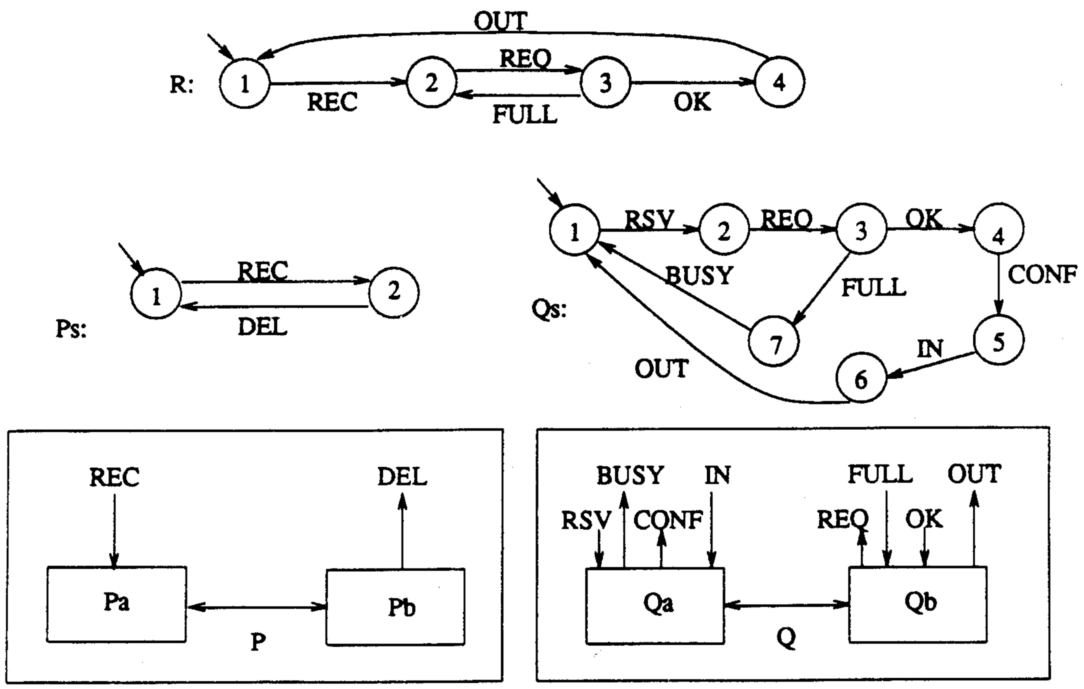


Figure 61: Example 3: Service Specifications

Note that in Fig. 64, state [2, 2, 1] (also [1, 2, 7]) has an incoming null transition and also an incoming non-null transition. This situation does not meet the simple conditions described in Section 7 for the aforementioned state removal. In this case, the enhanced null transition removal algorithm (Jeng and Liu, 1995) must be used. The conversion service specification obtained after applying the enhanced null transition removal algorithm to the CFSM in Fig. 63 is shown in Fig. 65. Following through step 3 of the STS algorithm, a synchronizing transition set is obtained as follows:

virtual atomic transition V_1 with
synchronizing transition set S_1 = { DEL, RSV, CONF, IN }

The next step (step 4) of the algorithm is the major difference between the new model and the original STS method. Recall that in the original STS algorithm, the intermediate converter specification is derived by composition of peer protocol entities P_b and Q_a at step 4. In the new model, the converter entity C in part (b) of Fig. 59 is composed of peer protocol entity P_b and the entire protocol Q (as a single entity). Since the peer transitions of Q are internal to Q and not observable from C's standpoint, only the service transitions need to participate in the composition in this step. In fact, protocol Q is treated as an entity consisting only of service transitions; i.e., the participants of this step in the new model are P_b and Q_s instead of P_b and Q_a. The same coordinated composition operator $\odot$ defined in Section 7.4 is used to derive protocol entity C. In short, protocol entity C is derived as follows:

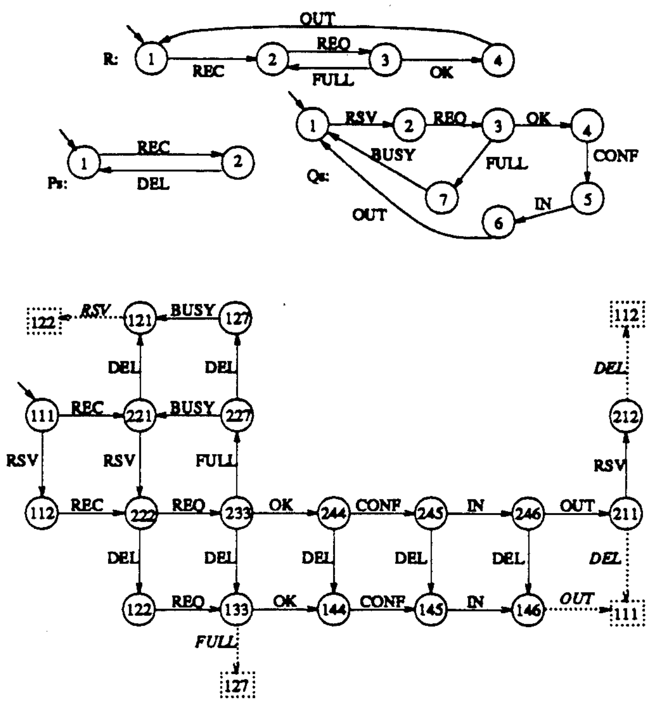


Figure 62: Example 3: Service System Graph G

converter entity $C = P_b \odot Q_s$

With the synchronizing transition set S_1 obtained in step 3, the result of applying the coordinated composition operator to P_b and Q_s is shown in Fig. 66. Note that Q_s, which is one of the given FSMs in the protocol conversion problem, replaced Q_a in this step (step 4). One may argue that Q_s contains service transitions from Q_a which are not needed in the final converter entity C specification and should not participate in the composition either. In answer to this argument, the service transitions from Q_a are needed for coordinating purposes in the coordinated composition operation. The service transitions of Q_b are synchronized with the peer transitions of P_b by the coordination of the service transitions from the synchronizing transition set which includes service transitions from P_b and Q_a. Therefore, the service transitions from Q_a serve as the coordinating agent in this step. It is in the next step (step 5) that the service transitions from Q_a are removed.

In the example, note that the synchronizing transition set contains more than one transition from Q_a that must be executed in a single virtual atomic transition; i.e., RSV, CONF, and IN from Q_a must be executed together with DEL from P_b. To reduce the complexity of the coordinated composition, Q_s can be simplified by combining the transition sequence RSV.REQ.OK.CONF.IN (which includes the service transitions from S_1) into a single atomic transition before the composition with P_b. In Fig. 66, Q'_s is the simplified service specification of protocol Q, which in turn participates with P_b in the coordinated composition operation to derive the converter entity C.

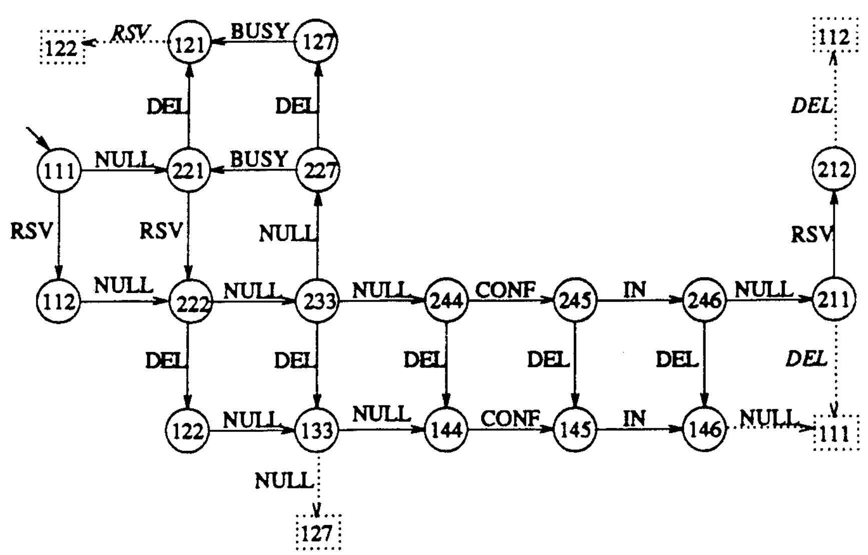


Figure 63: Example 3: Initial Projection

Finally, in the last step (step 5) of the algorithm, the virtual atomic transition V is expanded as in the original STS method. In addition, the service transitions from Q_a (RSV and BUSY) must be removed since only the service interface of Q_b needs to be supported by C and the service transitions RSV and BUSY are no longer needed after being used by the coordinated composition operation. In Fig. 67, the virtual atomic transition V is expanded and the service transitions from Q_a are labeled as NULL transitions. Again, the NULL transitions can be removed using the same NULL transition removal algorithm in polynomial time. The specification of the final converter entity C is shown in Fig. 68.

From the converter entity specification shown in Fig. 68, it can be verified that states [3, 7] and [1, 7] are redundant states. This indicates the need for further refinement of the modified STS algorithm. Due to the space limitation, the readers are referred to (Jeng and Liu, 1995) for the refinement.

10 Conclusions

In this article, various aspects of network interconnection, a rapidly growing area of research and practice in computer communications, have been described. There are a number of circumstances that lead to the growth in interconnecting separate networks together or to subdivide a large network into an interconnected set of smaller ones (Green, 1987):

- A single network can become so large that it exceeds architected limits on, for example, address space, or becomes unwieldy to manage in other respects.

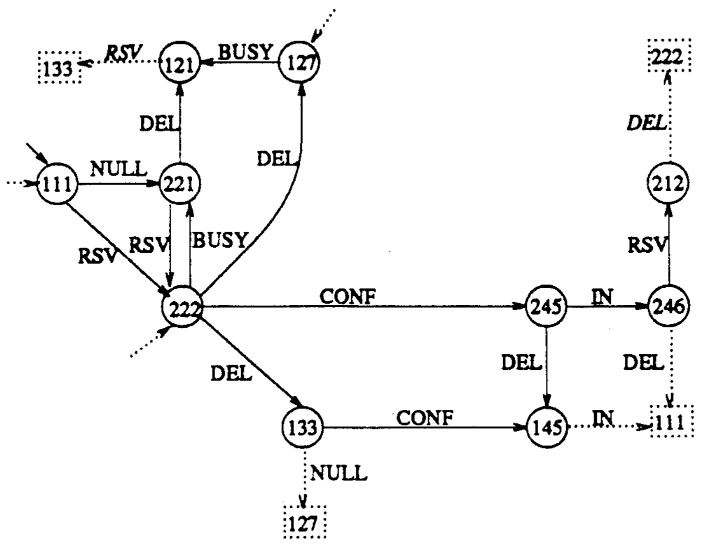


Figure 64: Example 3: Intermediate Projection Result

- Machines embedded in the networks of different enterprises may need to talk to one another.
- Stations on two local-area networks may need to talk to each other through a third wide area network.
- Partitioning according to geography may be required to save on line costs.
- Partitioning may be dictated by organizational subdivisions or by cost accounting or access control considerations.
- It may be desirable to put all the machines of different size ranges in different networks (e.g. all the PCs in one and all the mainframes in another).
- Analogously, it may be desirable to partition according to link speed (e.g. all the T1-connected machines in one network and all the voice grade modem-connected machines in another).

To facilitate the fast growing demand of network interconnection, various agencies and organizations have adopted internetworking standards, such as the ISO 8473 standard and DoD TCP/IP. Still, as technology advances, new requirements and demands are formed continuously, which in turn becomes the driving force for the development of new standards and/or better internetworking technologies.

The previous work on the network interconnection subject is extensive but scattered. Various (some may even be inconsistent) technologies exist and are being applied to different networks, which make the internetworking issues even more complex. In the first part of this article, various design issues, approaches and examples on the subject were examined. In terms of network interconnection,

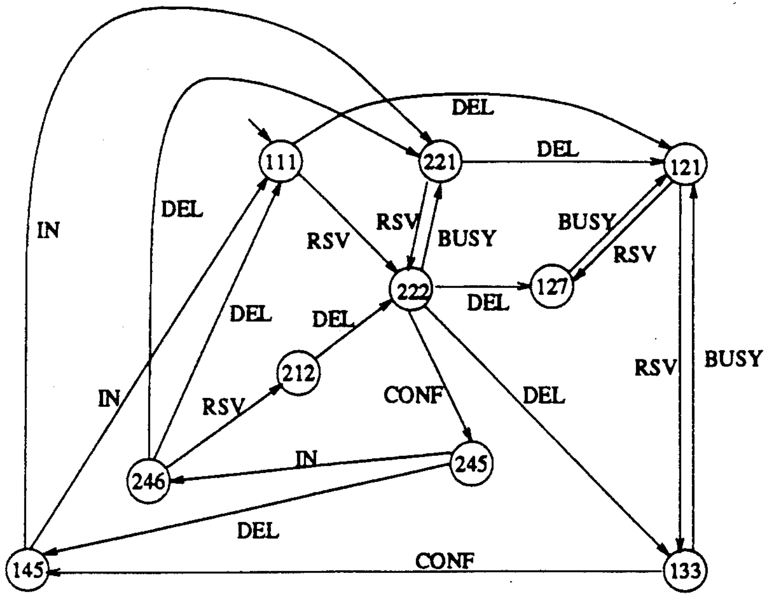


Figure 65: Example 3: Conversion Service Specification

while one solution to a problem may fit in a situation perfectly, it could fail to yield a satisfactory result in another. Consequently, many existing network interconnection examples use ad hoc solutions.

Ad hoc solutions are expensive and limited. Clearly, a systematic methodology is needed to alleviate the complexity of network interconnection, particularly in a heterogeneous network environment. A *formal methods to protocol conversion* is one of such methodologies. It targets toward one of the most important issues in network interconnection, *protocol mismatch*, which is a direct result of the diversity of the existing networks.

Protocol conversion has been an important area of research in *protocol engineering*. In general, a protocol conversion algorithm takes as input the specification of two target protocols and generates as output a converter specification for the networks to be interconnected.

In the second part of this article, the STS algorithm developed at The Ohio State University is introduced as the most powerful and versatile method for protocol conversion that has been developed so far. The STS algorithm has several advantages over other conversion methods reported in the literature, in terms of complexity, applicability, *correctness* properties, etc. It has been shown that the STS algorithms can be applied to protocols specified in the CFSM model (Section 7) and EFSM model (Section 8), and to multimedia networks as well (Section 9). Moreover, the STS algorithm has been coded using only 940 lines of language C statements (Jeng, 1995).

Since network interconnection and protocol mismatch are facts of life in today's fast advancing computer communication world, it is anticipated that protocol conversion will continue to be the

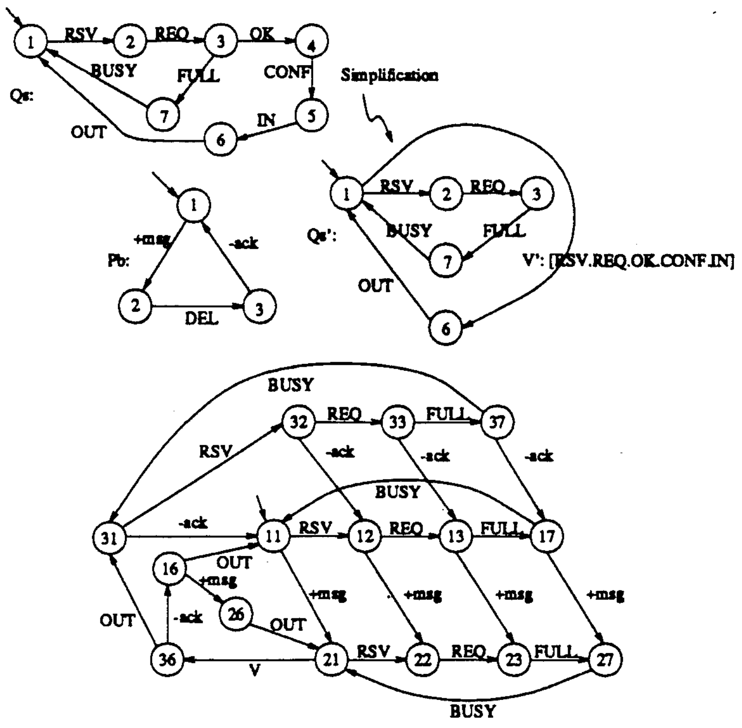


Figure 66: Example 3: Converter Entity C

emphasis of the network interconnection subject in the future.

ACKNOWLEDGEMENTS

I should like to express my appreciation to many people who have been involved with the protocol engineering project at The Ohio State University. Thanks are due to my former students Drs. J. Chang, I. Y. Chiou, W. S. Chen, H. W. Jeng, J. C. Shu and Y. W. Yao; the research reported in this article are mostly based on their dissertation research under my supervision. Other students currently involved with the project are Q. Guo, J. He, L. S. Kuo, Q. Liu, R. M. Rao, X. Shen, C. H. Wu and Y. Yang.

Special thanks are due to Dr. H. W. Jeng, who has assisted in collecting the references and in writing the article; without him the article might have never been completed.

Research reported herein has been supported by a series of contracts from U.S. Army CECOM and U.S. Army Research Office. My students and I are very grateful to Drs. Charles J. Graff and

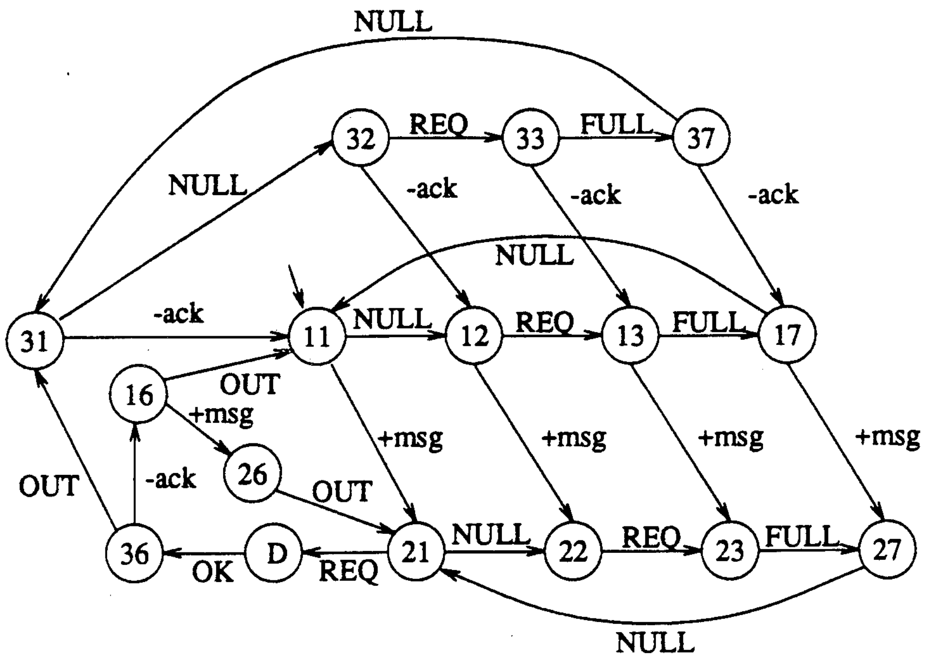


Figure 67: Example 3: Converter Entity C (V Expanded)

William A. Sander, the project monitors, for their continuing support and for their foresight in sponsoring the research project on protocol engineering.

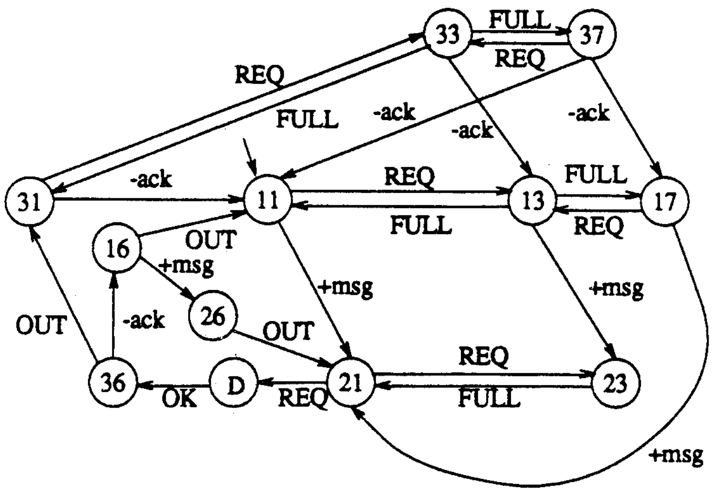


Figure 68: Example 3: Converter Entity C with NULL Transitions Removed

References

Adams, C., Burren, J., Cooper, C., and Girard, P. (1981). The interconnection of local area networks via a satellite network. In *Proc. NATO Adv. Study Inst.*, Bonas, France.

Ahmad, R. and Halsall, F. (1993). Interconnecting high-speed LANs and backbones. *IEEE Network*, pages 36–43.

Albanese, A., Degrandi, G., and Garrett, M. (1986). An architecture for transparent MAN/LAN gateways. In *Int. Conf. of Communication (ICC'86)*, pages 1255–1259, Toronto, Canada.

Antnakopoulos, T., Koutsonikos, J., and Makios, V. (1991). Design, implementation and performance analysis of an ETHERNET to LION gateway. *NATO ASI Series: High-Capacity Local and Metropolitan Area Networks*, F 72:455–469.

Baratz, A. and Jaffe, J. (1986). Establishing virtual circuits in large computer networks. *Computer Networks and ISDN Systems*, 12(1):27–37.

Beau, O. (1991). Introduction of ATM cross-connects in a broadband network: getting ready for B-ISDN. In *Telecom 1991 Forum*.

Benjamin, J., Hess, M., Weingarten, R., and Wheeler, W. (1983). Interconnecting SNA networks. *IBM System Journal*, 22(4):344–366.

Biocca, A., Freschi, G., Forcina, A., and Melen, R. (1990). Architectural issues in the interoperability between MANs and the ATM network. In *Proc. ISS'90*, Stockholm, Sweden.

Boggs, D., Shoch, J., Taft, E., and Metcalfe, R. (1980). Pup: an internetwork architecture. *IEEE Trans. on Communications*, 28(4):612–623.

Borgonovo, F. (1987). ExpressMAN: exploiting traffic locality in expressnet. *IEEE Journal on Selected Areas of Communications*, SAC-5(9):1436–1443.

Cain, E. and Cerf, V. (1983). The DoD internet architecture model. *Computer Networks*, 7:307–318.

Calvert, K. L. and Lam, S. S. (1987). An exercise in deriving a protocol converter. In *Proc. ACM SIGCOMM 1987 Symposium*, pages 151–160, Stowe, Vermont.

Calvert, K. L. and Lam, S. S. (1989). Deriving a protocol converter: a top-down method. In *Proc. ACM SIGCOMM 1989 Symposium*, pages 247–258, Austin, Texas.

Calvert, K. L. and Lam, S. S. (1990a). Adaptors for protocol conversion. In *Proc. IEEE INFOCOM 1990*, pages 552–560, San Francisco.

Calvert, K. L. and Lam, S. S. (1990b). Formal methods for protocol conversion. *IEEE Journal on Selected Areas in Communications*, 8(1):127–142.

Casoria, A. (1985). Interconnections and services integration in public and private networks for office automation. In *Proc. IEEE INFOCOM 1985*, pages 56–69.

CCITT (1978). Recommendation X.121 - int. numbering plan for public data networks. *Study Group VII, 76-E, Int. Telecommuncation Union.*

Cerf, V. (1979). Internet addressing and naming in a tactical environment. *IEN 110.*

Cerf, V. and Kahn, R. (1974). A protocol for packet network interconnection. *IEEE Tran. Comm.*, COM-22(5):637–648.

Cerf, V. and Kitstein, P. (1978). Issues in packet-network interconnection. In *Proc. IEEE*, volume 66, pages 1386–1408.

Chang, J. and Liu, M. T. (1990a). An approach to protocol complementation for interworking. In *Proc. IEEE ICSI 1990*, pages 205–211, Morristown, New Jersey.

Chang, J. and Liu, M. T. (1990b). Using protocol validation technique to solve protocol conversion problems. In *Proc. 9th IPCCC 1990*, pages 539–546, Scottsdale, Arizona.

Cheng, Y. and Robertazzi, T. (1987). Annotated bibliography of local communication system interconnection. *IEEE JSAC*, SAC-5:1492–1499.

Chew, E. (1983). Internetworking: local area network and public networks. In *Proc. 10th Australian Computer Conf.*, pages 586–600.

Chiou, I. Y. and Liu, M. T. (1986). GATENET: A voice/data internet transport system. In *Proc. IEEE INFOCOM 1986*, pages 39–46, Miami, FL.

Chou, W. (1985). *Computer Communications, Volumn II, Systems and Applications.* Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

Clapp, G. (1991). LAN interconnection across SMDS. *IEEE Network Mag.*, 5(5):25–32.

Clipsham, W., Glave, F., and Narraway, M. (1976). Datapac network overview. In *Proc. 3rd Int. Conf. Computer Communications*, pages 131–136.

Comer, D. (1995). *Internetworking with TCP/IP, Vol. I, Third Edition.* Prentice Hall, Englewood Cliffs, NJ.

Cotton, I. W. (1977). *Computer Network Interconnection: Problems and Prospects.* U.S. Department of Commerce, National Bureau of Standards, Washington, D. C.

Cotton, I. W. (1978). Computer network interconnection. *Computer Networks*, 12(1):25–34.

Crocetti, P., Fratta, L., Gerla, M., Marsiglia, M., and Romano, D. (1993). Interconnection of LAN/MANs through SMDS on top of an ATM network. *Computer Communications*, 16(2):83–92.

Crocetti, P., Galassi, G., and Gerla, M. (1991). Bandwidth advertising for MAN/LAN connectionless internetting. In *Proc. INFORCOM*, pages 186–193, Bal Harbour, FL.

Danet, A., Despres, R., Rest, A. L., Pichon, G., and Ritzenthaler, S. (1976). The french public packet switching service, the TRANSPAC network. In *Proc. 3rd Int. Conf. Computer Communications*, pages 251–269.

Davies, G. (1976). EURONET project. In *Proc. 3rd Int. Conf. Computer Communications*, pages 229–239.

Dixon, R. and Pitt, D. (1988). Addressing, brudging and source routing. *IEEE Network*, 2(1):25–32.

Doll, D. (1974). Telecommunications Turbulence and the Computer Network Evolution. *IEEE Computer*, pages 13–22.

Dowd, P. and Jabbour, K. (1987). A unified approach to local area network interconnection. *IEEE Journal on Selected Areas of Communications*, SAC-5(9):1418–1425.

Dunn, R. (1984). Local area network gateways to wide area networks. *Brit. Telecom Technol. J.*, 2(3):28–32.

Dupuy, S., Tawbi, W., and Horlait, E. (1992). Protocols for high-speed multimedia communications networks. *Computer Communications*, 15(6):349–358.

Estrin, J. and Carrico, B. (1982). Gateways promise to link local networks into hybrid systems. *Electronics*, pages 13–22.

Francois, P. and Potocki, A. (1983). Some methods for providing OSI transport in SNA. *IBM Journal of Research & Development*, 27(5):452–463.

Fraser, A. (1974). SPYDER - a data communications experiment. Technical Report Computer Sci. 23, Bell Labs.

Fratta, L., Borgonovo, F., and Togagi, F. (1981). The expressnet: a local area communication network integrating voice and data. In *Proc. Int. Conf. Performance of Data Communications, Systems and Applications*, pages 77–88, Paris, France.

Gemmell, J. and Christodoulakis, S. (1992). Principles of delay-sensitive multimedia data storage and retrieval. *ACM Transactions on Information Systems*, 10(1):51–90.

Gerla, M., Tai, T., Monteiro, J., and Gallassi, G. (1991). Interconnecting LANs and MANs to ATM. In *Proc. 16th LCN Conf.*, pages 54–62, Minneapolis.

Gien, M. and Zimmermann, H. (1979). Design principles for network interconnection. In *Proc. 6th Data Communications Symp.*, pages 109–119.

Green, P. (1986). Protocol conversion. *IEEE Transactions on Communications*, COM-34(3):257–668.

Green, P. (1987). Editor's notes. In *Network Interconnection and Protocol Conversion*, pages 7–8. IEEE Press.

Groenback, I. (1984). The TCP and ISO transport service - a brief description and comparison. *SHAPE Technical Center*, pages 898–913.

Groenback, I. (1986). Conversion between the TCP and ISO transport protocols as a method of achieving interoperability between data communication systems. *IEEE JSAC*, SAC-4(2):288–296.

Halsall, F. (1992). *Data Communications, Computer Networks and Open Systems, Third Edition.* Addison-Wesley, Workingham, England.

Haltzer, J., Reed, D., and Clark, D. (1981). Source routing for campus-wide internet transport. In *Local Networks for Computer Communications.* Amsterdam: North Holland.

Hawe, B., Kirby, A., and Stewart, B. (1984). Transparent interconnection of local area networks with bridge. *Journal of Telecommunication Networks*, 3(2):116–130.

Heinanen, J. (1992). Frame realy as a multiprotocol backbone interface. *Computer Networks and ISDN Systems*, 25:363–369.

Helsel, F. and Spadafora, A. (1976). Siemens system EDS - a new stored program crotrolled switching system for telex and data networks. In *Proc. 3rd Int. Conf. Computer Communications*, pages 51–55.

Hirsch, P. (1974). SITA rating a packet-switched network. *Datamation*, 20:60–63.

Hoepner, P. (1992). synchronizing the presentation of multimedia objects. *Computer Communications*, 15(9):557–564.

Hopcroft, J. E. and Ullman, J. D. (1979). *Introduction to Automata Theory, Language and Computation.* Addison-Wesley Publishing Company, Reading, Massachusetts.

IEEE (1985). *IEEE peoject 802 - local and metropolitan area network standards.* IEEE Standard 802.1 (D): MAC Bridges.

IEEE (1989). *Draft of proposed IEEE standard 802.1 MAC bridges.* Draft P802.1d/D8.

Isaku, S. and Ishukura, M. (1990). ATM network architecture for supporting the connectionless service. In *Proc. INFOCOM 90*, pages 122–131, San Francisco, CA.

ITU (1991). *Frame mode bearer service (FMBS).* ITU, Geneva.

Jeng, H. J. (1995). *Synchronizing Transition Set Approach to Protocol Conversion.* PhD thesis, The Ohio State University.

Jeng, H. J. and Liu, M. T. (1992). From service specification to protocol converter: a synchronizing transition set approach. Technical Report No. OSU-CISRC-3/92-TR-10, The Ohio State University, Columbus, Ohio.

Jeng, H. J. and Liu, M. T. (1993). Protocol conversion: synchronizing transition set approach in EFSM model. Technical Report No. OSU-CISRC-1/93-TR-2, The Ohio State University, Columbus, Ohio.

Jeng, H. J. and Liu, M. T. (1995). Protocol conversion in multimedia networks: simulation and algorithm. *SIMULATION*, 64(1):51–60.

Kahn, R. (1975). The organization of computer resources in a packet radio network. In *Proc. Nat. Computer Conf.*, pages 177–186.

Karp, P. (1973). Origin, development and current status of the ARPANET. In *Proc. COMPCON 73*, pages 49–52.

Kelekar, S. G. and Hart, G. W. (1993). Synthesis of protocols and protocol converters using the submodule construction approach. Technical Report No. CU/CTR/TR 322-93-01, Columbia University, New York.

Kummer, P., Linge, T., and Ball, E. (1987). A protocol-less scheme for bridging between IEEE 802 local area networks. *Computer Networks and ISDN Systems*, 12(2):81–88.

Kung, H. (1992). Gigabit local area networks: a systems perspective. *IEEE Communications Mag.*, 30(4):90–101.

Lam, S. S. (1986). Protocol conversion - correctness problems. In *Proc. ACM SIGCOMM 1986 Symposium*, pages 19–28, Stowe, Vermont.

Lam, S. S. (1988). Protocol conversion. *IEEE Transactions on Software Engineering*, SE-14(3):353–362.

Lam, S. S. and Shankar, A. U. (1984). Protocol verification via projections. *IEEE Trans. Software Engineering*, SE-10(4):325–342.

Landegem, T. V. and Peschi, R. (1991). Managing a connectionless virtual overlay network on top of ATM. In *Proc. ICC'91*, pages 122–131, Denver, CO.

Lapidus, G. (1976). SWIFT network. *Data Communications*, 5(5):20–24.

Larsson, T. (1976). A public data network in the nordic countries. In *Proc. 3rd Int. Conf. Computer Communications*, pages 246–250.

Liu, M. T. (1978). Distributed loop computer networks. In *Advances in Computers*, volume 17, pages 163–221. Academic Press, New York.

Liu, M. T. (1989). Protocol engineering. In *Advances in Computers*, volume 27, pages 79–195. Academic Press, New York.

Liu, M. T. (1990). Protocol conversion. In *Proc. International Computer Symposium*, pages 82–92, Hsinchu, Taiwan.

Lloyd, D. and Kirstein, P. (1975). Alternative approach to the interconnection of computer networks. *Eurocomp*, pages 499–518.

Luvison, A., Roullet, G., and Toft, F. (1987). The LION project: a status report. In *Proc. 4th Annual ESPRIT Conf.*, pages 1477–1489, Brussels, Belgium.

Mairs, C. (1990). Interworking with LU type 6.2. In *Networks for the 1990s*, pages 122–131. Online Publication, London.

Marchall, G. (1983). Bridge link lANs. *System and Software*, pages 122–131.

Maxemchuk, N. (1985). Regular mesh topologies in local and metropolitan area networks. *AT&T Tech. J.*, 64:1659–1685.

McConnell, J. (1988). *Internetworking Computer Systems.* Prentice Hall, Englewood Cliffs, New Jersey.

Merlin, P. M. and Bochmann, G. V. (1983). On the construction of submodule specifications and communication protocols. *ACM Tran. on Programming Languages and Systems*, 5(1):1–25.

Metcalfe, R. and Boggs, D. (1976). ETHERNET: distributed packet switching for local computer networks. *Comm. of ACM*, 19(7):395–404.

Moeller, E., Scheller, A., and Schurmann, G. (1990). Distributed multimedia information handling. *Computer Communications*, 13(8):232–242.

Mongiovi, L., Farrel, M., and Trecordi, V. (1991). A proposal for interconnecting FDDI networks through B-ISDN. In *Proc. INFOCOM 91*, pages 122–131, Bal Harbour, FL.

Munafo, M., Feri, F., Cioffari, C., and Vasco, A. (1993). Analysis of the spanning tree and source routing LAN interconnection schemes. In *Local Area Network Interconnection*, pages 161–182. Plenum Press, New York.

Nakamura, R., Ishino, F., Sasaoka, M., and Nakamura, M. (1976). Some design aspects of a public switched network. In *Proc. 3rd Int. Conf. Computer Communications*, pages 317–322.

Okumura, K. (1986). A formal protocol conversion method. In *Proc. ACM SIGCOMM 1986 Symposium*, pages 30–38, Stowe, Vermont.

Okumura, K. (1990). Generation of proper adaptors and converters from a formal service specification. In *Proc. IEEE INFOCOM 1990*, pages 564–571, San Francisco.

Periman, R. (1980). Flying packet radios and network partitions. *IEN 146*, pages 485–500.

Periman, R. (1982). Hierarchical networks and the subnetwork partition problem. In *Proc. 15th Int. Conf. on System Science*, pages 122–131.

Perlman, R. (1992). *Interconnection: Bridges and Routers.* Addison-Wesley Publishing Co., Reading, MA.

Peyravian, M. and Lea, C. (1993). Construction of protocol converters using formal methods. *Computer Communications*, 16(4):215–228.

Pitt, D. and k. Sy (1986). Address-based and non-address-based routing schemes for interconnected local area networks. In *Local Area and Multiple Access Networks*, pages 155–166. Computer Science, Rockville, MD.

Pitt, D. and Winkler, J. (1987). Table-free bridging. *IEEE Journal on Selected Areas in Communications*, SAC-5(9):1454–1462.

Postel, J. (1980). Internetwork protocol approaches. *IEEE Transactions on Communications*, 28(4):604–611.

Pouzin, L. (1973). Presentation and major design aspects of the CYCLEDES computer network. In *Proc. 3rd Data Communications Symp.*, pages 80–85.

Prycker, M. (1990). *Asynchronous Transfer Mode: Solution for B-ISDN.* Ellis Horwood, Prentice Hall, London.

Prycker, M. (1992). ATM technology: a backbone for high speed computer networking. *Computer Networks and ISDN Systems*, 25:357–362.

Ramanathm, S. and Rangan, P. (1993). Adaptive feedback techniques for synchronized multimedia retrieval over integrated networks. *IEEE/ACM Transactions on Networking*, 1(2):246–260.

Rangan, P., Vin, H., and Ramanathm, S. (1993). Communication architectures and algorithms for media mixing in multimedia conferences. *IEEE/ACM Transactions on Networking*, 1(1):20–30.

Rinde, J. (1976). TYMNET: an alternative to packet switching technology. In *Proc. 3rd Int. Conf. Computer Communications*, pages 268–273.

Roberts, L. and Wessler, B. (1973). The ARPA network. In *Computer-Communications Networks*, pages 485–500. NJ: Prentice-Hall.

Roberts, L. G. (1975). Telenet: principles and practice. In *Proc. Eur. Computing Conf. Communication Networks*, pages 315–329.

Rodriguez, J. M. (1988). An X.PC/TCP protocol translator. In *Proc. IEEE INFOCOM 1988*, pages 308–313.

Roffinella, D., Trinchero, C., and Freschi, G. (1987). Interworking solutions for a two-level integrated service local area network. *IEEE Journal on Selected Areas of Communications*, SAC-5(9):1444–1453.

Rose, M. T. and Cass, D. E. (1987). OSI transport service on top of TCP. *Computer Networks and ISDN Systems*, 12:159–173.

Ryder, K. (1983). An experimental address space isolation techinque for SNA networks. *IBM System . Journal*, 22(4):367–386.

Sanders, R. and Vinton, C. (1976). Compatibility or chaos in communications. *Datamation*, pages 50–55.

Schoch, J. (1979). Packet fragmentation in inter-network protocols. *Computer Networks*, 3(1):3–9.

Schodl, W., Briem, U., Kroner, H., and Theimer, T. (1993). Bandwidth allocation mechanism for LAN/MAN interworking with an ATM network. *Computer Communications*, 16(2):93–99.

Sharon, O. and Segall, A. (1993). A simple scheme for slot reuse without latency for a dual bus configuration. *IEEE/ACM Transactions on Networking*, 1(1):96–104.

Shoch, J., Cohen, D., and Taft, E. (1980). Mutual encapsulation of internetwork protocols. In *IEEE Symp. on Trends and Applications: Computer Network Protocols*, pages 1–11.

Shu, J. C. and Liu, M. T. (1989). A synchronization model for protocol conversion. In *Proc. IEEE INFOCOM 1989*, pages 276–284, Ottawa, Canada.

Shu, J. C. and Liu, M. T. (1990). Protocol conversion between complex protocols. In *Proc. 9th IPCCC*, pages 584–590, Phoenix, AZ.

Shu, J. C. and Liu, M. T. (1991). An approach to indirect protocol conversion. *Computer Networks and ISDN Systems*, 21(2):93–108.

Stallings, W. (1994). *Data and Computer Communications, Fourth Edition*. Macmillan Publishing Company, New York.

Stockman, B. (1993). Global connectivity - the global internet exchange (GIX). *Computer Networks and ISDN Systems*, 26:297–303.

Sundstrom, R. and Schultz, G. (1980). SNA's first six years: 1974-1980. In *Proc. Fifth International Conference on Computer Communication*, pages 578–585, Atlanta, GA.

Sunshine, C. (1977). Source routing in computer networks. *ACM Computer Communications Rev.*, 7(1):122–131.

Sunshine, C. (1982). Addressing problems in multi-network system. In *Proc. IEEE INFOCOM, 1982*, pages 12–18, Las Vegas, NV.

Sunshine, C. A. (1983). Experience with automated protocol verification. In *Proc. IFIP Int. Workshop on PSTV. 3rd*, pages 229–236.

Sy, K., Shiobara, M., Yamaguchi, M., Kobayashi, Y., Shukuya, S., and Tomatsu, T. (1987). OSI-SNA interconnections. *IBM System Journal*, 26(2):157–173.

Sze, D. (1985). A metropolitan area network. *IEEE JSAC*, SAC-3:815–824.

Thornton, J. and Christenson, G. (1983). Hyperchannel network links. *IEEE Computer*, pages 122–131.

Tirtaamadja, E. and Palmer, R. (1990). The application of virtual paths to the interconnection of IEEE 802.6 metropolitan area networks. In *Proc. ISS 90*, pages 22–30, Stockholm, Sweden.

Tobagi, F., Borgonovo, F., and Fratta, L. (1983). Expressnet: a high performance integrated services local area network. *IEEE JSAC*, SAC-1:898–913.

Turner, C. and Peterson, L. (1992). Image transfer: an end-to-end design. In *Proc. SIGCOMM 92*, pages 258–268, Baltimore, Maryland.

Wakayama, H., Kobayashi, Y., Ohara, Y., and Shiobara, M. (1985). Application of OSI protocols as intermediary for DCNA-SNA networks interconnection. In *Proc. III Conf. on Introduction of Open System Connections Standards*, pages 46–57.

Weir, D., Holmblad, J., and Rothberg, A. (1980). An X.75 based network architecture. In *Proc. 5th Int. Conf. on Computer Communications*, pages 741–750.

Wong, J., Vernon, A., and Field, J. (1987). Evaluation of a path-finding algorithm for interconnected local area networks. *J. Selected Areas in Communications*, SAC-5(9):1463–1470.

Yang, O. and Mark, J. (1986). Design issues in metropolitan area networks. In *Int. Conf. of Communications (ICC'86)*, Toronto, Canada.

Yao, Y. W., Chen, W. S., and Liu, M. T. (1990a). A modular approach to constructing protocol converters. In *Proc. IEEE INFOCOM '90*, pages 572–579, San Francisco.

Yao, Y. W., Chen, W. S., and Liu, M. T. (1990b). A parallel model for constructing protocol converters. In *Proc. IEEE GLOBECOM '90*, San Diego, California.

Yao, Y. W. and Liu, M. T. (1992). Constructing protocol converters from service specification. In *Proc. IEEE ICDCS-12*, pages 344–351, Yokohama, Japan.

Yin, N. and Hluchyj, M. (1991). A dynamic rate control mechanism for source coded traffic in a fast packet network. *IEEE Journal on Selected Areas in Communications*, 9(7):1003–1991.

Zoline, K. and Lidinsky, W. (1985). An approach for interconnecting SNA and XNS networks. In *Proc. 9th Data Communications Symp.*, pages 184–198.